VISUALIZING AND MODELING PARTIAL INCOMPLETE RANKING DATA

A Dissertation
Presented to
The Academic Faculty

by

Mingxuan Sun

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
Computer Science

College of Computing
Georgia Institute of Technology
December 2012

VISUALIZING AND MODELING PARTIAL INCOMPLETE RANKING DATA

Approved by:

Professor Guy Lebanon, Advisor
College of Computing
Georgia Institute of Technology

Doctor Kevyn Collins-Thompson
Microsoft Research

Professor Hongyuan Zha
College of Computing
Georgia Institute of Technology

Professor Alexander Gray
College of Computing
Georgia Institute of Technology

Professor Charles L. Isbell
College of Computing
Georgia Institute of Technology

Date Approved: 22 August 2012

To my parents and to my husband.

ACKNOWLEDGEMENTS

I am deeply grateful to my advisor Guy Lebanon for his guidance throughout the years of my study. Without his impressive ideas, extraordinary experiences and constant supports, this work is simply not possible. Guy directed me into the field of machine learning and kept me focused in my research consistently. I have been inspired a lot from his unique ways of approaching novel problems and his abilities to solve different problems with solid math. He also helped me to improve my academic writing and oral presentation significantly.

I would like to thank all members of my thesis committee: Hongyuan Zha, Kevyn Collins-Thompson, Charles L. Isbell and Alexander Gray. They have provided valuable feedback on my thesis and helped me to enhance the quality of my work significantly. I am grateful to Hongyuan and Kevyn for collaborating on information retrieval projects with valuable insights and discussions. I also benefit a lot from Charles's machine learning class and gain fundamental knowledge. Alex's group has always been a valuable source of discussing and exchanging research ideas in machine learning.

Additionally, there are a number of lab mates providing me useful insights that influence the work. In particular, I am grateful to Paul kidwell and Yi Mao for providing useful discussions and lessons from their own experiences. I am also grateful to Fuxin Li, Joonseok Lee, Seungyeon Kim, and Ke Zhou for valuable discussions and collaborations. Thanks to Joshua Dillon, Krishnakumar Balasubramanian, Da Kuang, Jiang Bian, Shuanghong Yang, Dongyeol Lee, Jingu Kim, and all members from the Fastlab, with whom I spent my most fruitful years at Tech, for providing discussions and fun in the lab.

Finally, I would like to thank my parents for their unquestioning support throughout my life. They helped me to shape positive attitudes in both work and life. Most importantly, I would like to thank my husband Xiangwei for his unconditional love and tolerance of my impatience, without which I could never made this far.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Analyzing ranking data is an essential component in a wide range of important applications including web-search and recommendation systems. Rankings are difficult to visualize or model due to the computational difficulties associated with the large number of items. On the other hand, partial or incomplete rankings induce more difficulties since approaches that adapt well to typical types of rankings cannot apply generally to all types. While analyzing ranking data has a long history in statistics, construction of an efficient framework to analyze incomplete ranking data (with or without ties) is currently an open problem.

This thesis addresses the problem of scalability for visualizing and modeling partial incomplete rankings. In particular, we propose a distance measure for top-k rankings with the following three properties: (1) metric, (2) emphasis on top ranks, and (3) computational efficiency. Given the distance measure, the data can be projected into a low dimensional continuous vector space via multi-dimensional scaling (MDS) for easy visualization. We further propose a non-parametric model for estimating distributions of partial incomplete rankings. For the non-parametric estimator, we use a triangular kernel that is a direct analogue of the Euclidean triangular kernel. The computational difficulties for large n are simplified using combinatorial properties and generating functions associated with symmetric groups. We show that our estimator is computational efficient for rankings of arbitrary incompleteness and tie structure. Moreover, we propose an efficient learning algorithm to construct a preference elicitation system from partial incomplete rankings, which can be used to solve the cold-start problems in ranking recommendations.

The proposed approaches are examined in experiments with real search engine

and movie recommendation data. The visualization of top-k rankings is highly effective in summarizing and analyzing insights on search engine dissimilarities, query manipulation diversities, and query intent ambiguities. The partial ranking estimator is successfully applied to collaborative filtering recommendation systems, where the conditional probability estimation is naturally suited for tasks such as rank prediction and association rule discovery. The preference elicitation system constructed by our learning algorithm achieves high prediction accuracy with low user interaction time for cold-start recommendations.

CHAPTER I

INTRODUCTION

Rankings occur in a wide range of common life. The scenarios include political election where votes rank top candidates, searching engines outputting top-k web pages, and customer preferences over different products. In their simplest form, rankings arise from a set of m raters ranking a set of n items. A fully ordered data is a set of permutations mapping abstract items to their rankings. In many cases, expecting every rater to judge every items is unrealistic and rankings arrive in forms different from full orderings. For example, observed from the American Psychological Association (APA) election data, voters asked to rank all candidates often rank only a few top candidates. In other words, the rankings are in the form of partial rankings. In practical recommendation systems, the number of indexed items, which may be books, movies, or songs, is relatively high in the order of $10^3 - 10^4$ at least. However, each user observes only a small subset of the items and provides ratings from a scale of 1 (bad) to 5 (good). As a result, the preference rankings are partial incomplete rankings.

Analyzing rankings has always been an essential component in statistics. For a long history, statisticians have developed methods or models from different aspects including distance measure, effective visualization, and probability modeling and reasoning. Popular distance measures for permutations includes Kendall's tau, Spearman's rho, Footrule, Ulam's distance, and Cayley's distance [20]. A popular visualization tool usually includes the permutation polytope by plotting the rank vectors in Euclidean space. A lower dimensional projection of the polytope is suggested by [16, 93] for items of large sizes. Early attempts of the modeling focusing

on simple generative models include the Thurstone model [94] and the Babington Smith model [87]. A special case of the Babington Smith model is the Mallows [63], which appears as an analogue of the Gaussian distribution. Since 1980s, statisticians have put a great amount of efforts to broaden the views. Highlights include group representations with analysis of variance models by Diaconis [23, 24], a unified view for partial rankings by Critchlow [20], and multistage models extending the Mallows by [27, 28], all of which provide us wealthy information and insights.

Over the last past decades, ranking data are involved frequently in challenging machine learning problems. In the information retrieval fusion problems [30, 98, 64, 17, 4], the ranking data are typically a ranked list of web pages output by different search engines. The task of rank fusion is to combine the rankings to obtain a "meta-search" engine of higher accuracy. In multi-object tracking systems, the goal is to managing multiple trails with assignment of each object to a trail. The track-to-identity assignment is formed as an inference problem on the group of permutations of objects. Exploration of rankings is also important in preference leaning and rule discovery in recommendation systems. For example in the movie recommendation, ranking analysis reveals important information such as some movies being more popular than the others, favoring one movie indicating low preference of another, and clusters representing users of different tastes.

Rankings are difficult to visualize or model due to the computational difficulties when the number of items is large. The probability space of n items is of size $n!$. Computations involving naive enumeration or summation require a time complexity of $O(n!)$. In fact, most of the data analyzed in the literature are limited to a small number of items such as the American Psychological Association (APA) election data where voters rank at most five candidates. Consequently, approximations are adopted to reduce the computational complexity in some approaches. On the other hand, partial or incomplete rankings induce more difficulties for modeling since models may

adapt well to some types but not the others.

Although a significant amount of work emerges in both statistical and machine learning literature, existing algorithms do not address the scalability issues of visualizing and modeling all types of rankings.

1.1 Thesis Statement

This thesis addresses problems of scalability for visualizing and modeling partial incomplete rankings. In particular, we present a new distance measure for partial rankings with the following three properties: (1) metric, (2) emphasis on top ranks, and (3) computational efficiency. The distance in conjunction with multi-dimensional scaling is effective for visualizing partial rankings. We further propose an efficient non-parametric model with triangular smoothing for estimating distributions from partial incomplete rankings. Moreover, we propose a learning algorithm to construct a decision tree for preference elicitation from partial incomplete rankings.

The thesis consists of four claims that I will defend in the dissertation.

1. The weighed Hoeffding distance for top-k rankings is computational efficient and has several advantages over conventional dissimilarity measures.

2. The weighted Hoeffding distance in conjunction with multi-dimensional scaling is effective for visualizing partial rankings.

3. The non-parametric model using triangular smoothing is efficient for modeling partial incomplete rankings.

4. The constructed decision tree for preference elicitation achieves high prediction accuracy with relatively low user interaction time.

In the reminder of the section, we provide brief reasoning that supports the claims.

1.2 Partial Incomplete Rankings

Partial Incomplete Rankings appear frequently in important applications such as search engines and recommendations systems. Partial (tied) rankings are a partition of disjoint subsets such that all items in one subset are preferred to all items in another but no information is provided concerning the relative preference of the items in the same set. Incomplete rankings with missing items occur when raters omit certain items from their preference information altogether. This case is very common in situations involving a large number of items.

Existing algorithms may work well for some types but not others. It is difficult to directly visualize and posit a coherent probabilistic model for incomplete tied data. In the following sections, we will outline the arguments to support thesis claims in terms of visualizing top-k rankings, modeling partial incomplete rankings and preference elicitation.

1.2.1 Visualization

One typical example of top-k rankings is the search engine output, which is a total ordering on the top k items with the rest items tied at bottom. A feature of top-k rankings is that top items are more important than bottom items. Substantial research on the interaction between users and search engines [35, 48] shows that user attentions drop quickly from top to bottom ranks.

An effective visualization of partial ranking of large items is needed. For a small size of items, a permutations polytope [102, 93, 5] is a sufficient tool, which is a convex hull in $\mathbb{R}^n$ space with $n!$ vertices each corresponding to a permutation. Two vertices are connected by an edge if the Kendall's tau distance is 1, which means the permutations differ only by transposing two adjacent items. The permutation polytope for 4 items is displayed in Figure 1 where it is embedded in $\mathbb{R}^3$. To reveal the probability distribution, it is suggest by [16] drawing a sphere at each vertex with

radius proportional to the probability of corresponding permutation. However, for rankings on medium or large size of items, the use of the polytope for visualization purposes is rather limited. Alternatively, an effective approach for visualizing high dimensional ranking data is to reduce the data on a lower dimensional space using stand dimensional reduction tools like principle component analysis (PCA) or multi-dimensional scaling (MDS).

In order to map the data from a high dimensional ranking space to a lower dimensional space for easy visualization, a distance metric for top-k rankings is necessary. In particular, a distance metric for top-k rankings needs to have several properties such as being symmetric, emphasizing on top ranks and computational efficiency. However, most of the existing metrics do not have all of the properties simultaneously. In fact, many proposed dissimilarities do not distinguish between disagreements at top and bottom ranks.

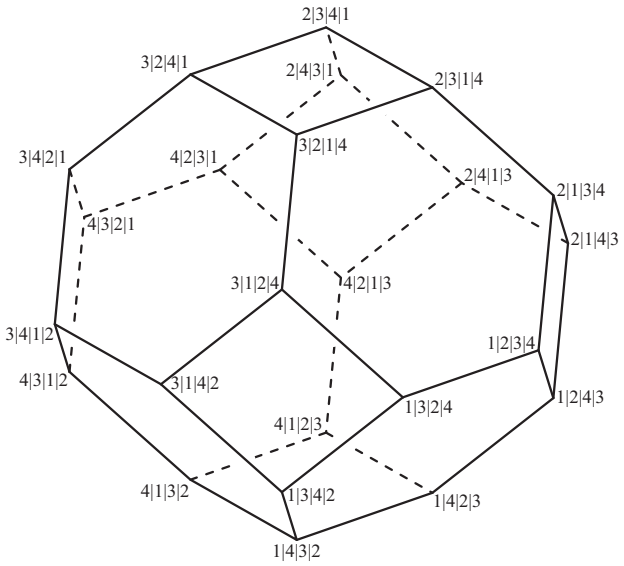


Figure 1: Permutation polytope for 4 objects represented in 3D space.

1.2.2 Probabilistic Modeling

Modeling ranking data is an essential component in a number of important applications. However, it has not been fully studied due to a number of challenges unique to ranking data. Rankings over a large number of items n reside in an extremely large space whose modeling often requires intractable computation. For example, maximum likelihood estimation may involve searching over $n!$ permutations in the ranking space. On the other hand, most of the existing models working well for fully ranked data are not applicable to incomplete data.

Many of the classic models such as [63] assume simple parametric forms for ranking distributions, which may break down as the number of items increases. Figure 2, appears also in [51], demonstrates how the number of modes and complexity increase with n. The density estimate of rankings using kernel smoothing is embedded in a two-dimensional space using multidimensional scaling. As n increases, the number of the modes increases and the density surface itself becomes less regular. Intuitively, different probability mass regions correspond to different types of judges. For example in movie preferences, probability modes may correspond to genre as fans of drama, action, or comedy having similar preferences.

A different class of estimators based on Fourier analysis on the symmetric group has been proposed by [23, 44]. These techniques vary in their statistical accuracy and computational efficiency with the latter becoming crucial for large n. Alternatively, a non-parametric model assuming a Mallows kernel has been proposed in [59] where efficient computational procedures are developed for tied rankings. Instead of using Mallows kernel, a triangular kernel is developed in [50] to apply the density estimation for incomplete rankings. However, the computational procedure is inefficient to estimate the probability of incomplete rankings of arbitrary structures.

To model incomplete rankings efficiently is a crucial problem in analyzing real

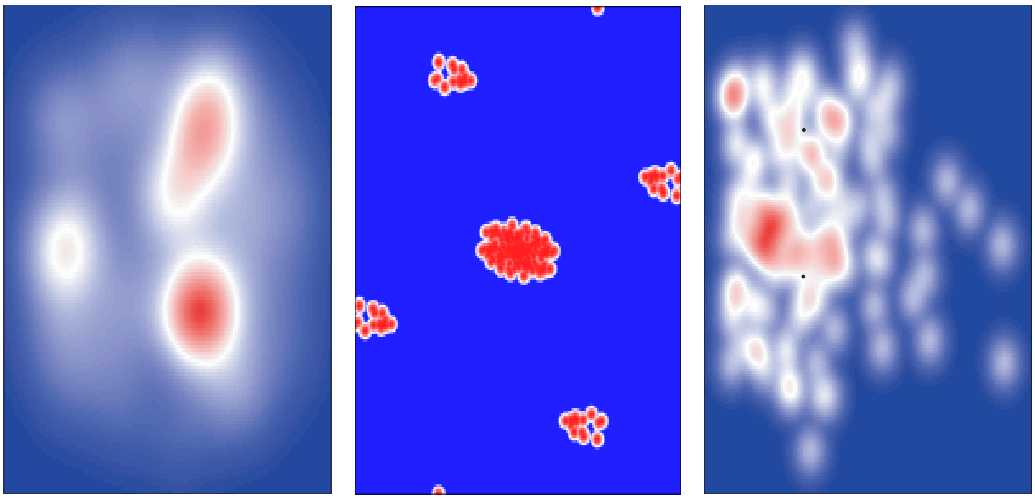


Figure 2: Heat map visualization of the density of ranking data using multidimensional scaling. The data sets are APA voting (left, $n = 5$), Jester (middle, $n = 100$), and EachMovie (right, $n = 1628$) data sets. None of these cases show a simple parametric form, and the complexity of the density increases with the number of items n. This motivates the use of non-parametric estimators for modeling preferences over a large number of items. The figure first appears in [51].

world data sets such as recommendation systems. The main difficulty is computational complexity: both parametric and non-parametric models require summation over sets that increase exponentially in the number of items n.

1.2.3 Preference Elicitation

Preference elicitation aims to soliciting the interests of new users to the recommender system and rapidly generates a good initial set of recommendations. A popular strategy is asking users to rate certain items through an initial interview in order to learn user preferences. A good interview is targeted at discovering user interests with minimum interactions, since users may grow tiresome and abandon the interview if too many questions are asked. Ideally, users should be queried adaptively in a sequential fashion, and multiple items should be offered for opinion solicitation at each trial.

State-of-the-art work [76, 104, 33] has championed decision trees [11] as a natural fit for the interviews of new users. They make quick decisions by only querying a few of the items from a large collection. Moreover, the query is chosen adaptively: in the

primitive form, the user is first asked one question, then assigned to different branches of the decision tree based on the user's previous responses. In the end, the decision is made based on the average of training user responses within each leaf node. The decision is learned from a training data of a collection of partial incomplete rankings.

However, a big challenge associated with preference learning in the interview process is the preference with missing items. In fact, most users only view a small amount of items. Many users have to repeatedly respond with "Unknown" multiple times before locating any item they know about. In this case, not only did the system gain little valuable information on users' interests, but the users also easily get bored and may opt out of the system prematurely. Algorithms focusing on asking multiple questions at each trial to increase the chance that a user would know some items are needed for a better cold-start recommender system.

1.3 Organization

The remainder of this dissertation is organized as follows. Related work is discussed in each of the remaining chapters. In Chapter 2, we introduce the basic concepts of ranking space, ranking types, and distance measures, and then present the weighted Hoeffding distance. In Chapter 3, we present a framework for visualizing partial ranking data for large n, which is effective and computational efficient, and then discuss an application of the framework in visualizing search results. In Chapter 4, we develop a computational efficient scheme for modeling partial incomplete rankings based on triangular smoothing, and then present an application of the framework in collaborative filtering and rule discovery in recommendation systems. In Chapter 5, we propose an efficient learning algorithm to construct a preference elicitation system from partial incomplete rankings, which can be used to solve the cold-start problems in ranking recommendations.

CHAPTER II

RANKING SPACE AND DISTANCE MEASURE

In this chapter, we review the basic concepts of ranking data including ranking space and ranking types. A ranking can be complete or incomplete. A ranking, whether it is complete or incomplete, can be with-ties (partial) or without-ties. Complete without-ties rankings correspond to permutations. We then review some traditional distance measures including Kendall's tau, Spearman's rho, Footrule, Ulam's distance, and Cayley's distance on permutations and the extension to partial incomplete rankings. In particular, we present a closed form formula to compute the expected distance of Kendall's tau for partial incomplete rankings. Most of the distance measures are not generalized to top-k rankings. Alternatively, we present the weighted Hoeffding distance with three nice properties and compare it with other distance measures.

2.1 Related Work

Popular distance measures between permutations are Kendall's tau, Spearman's rho, Footrule, Ulam's distance, and Cayley's distance [20]. Among them, Spearman's rho [25] and Footrule measure spatial distances between two vectors. Kendall's Tau [49] is the minimum number of transpositions of adjacent items to align two permutations. Instead of counting adjacent item swaps as in Kendall's, Carley's distance [13] is the minimum number of arbitrary pairwise interchanges. Another way to measure the disorder between two rankings is Ulam's distance [20], which is minimum work of removing and reinserting an item at places to make items in order. There are several ways to extend these permutation distances to partial incomplete rankings including Hausdorff distance [20] and expected distances [2, 3, 65].

One problem with many proposed dissimilarities, however, is that they do not

distinguish between disagreement at top ranks and at the bottom ranks. The feature of emphasizing top ranks is critical in measuring ranking dissimilarities especially for top-k rankings (e.g., search results). Substantial research on the interaction between users and search engines [35, 48] shows that users' attention drops quickly from top to bottom ranks[1]. Attempts to customize the distance includes [27], in which they considers stage-wise ranking processes that generalize Kendall's tau. This generalization, however, is not unique as it depends on the order in which the different ranking stages are selected. Rank correlation coefficient such as NDCG [46] adopts inverse logarithm function as the discount rank factor. However it is not symmetric and is intended as an evaluation measure against ground truth, not a comparison measure between rankings. Another example is the inverse measure [6] that emphasizes disagreement at top ranks, where the weight function decays linearly with the rank. However, it lacks discriminative power. We propose the weighed Hoeffding distance, which is computational efficient and has several advantages over conventional dissimilarity measures.

2.2 Complete or Incomplete and With-ties or Without-ties Rankings

Complete without-ties rankings correspond to permutations, also known as full rankings. The ranking space of n items has a discrete structure, which consists of $n!$ complete rankings. A full ranking π of the items $\{1, \ldots, n\}$ is a bijection to itself mapping items to ranks. Therefore, $\pi(3)$ is the rank given to item 3 and $\pi^{-1}(2)$ is the item that is assigned second rank. The set of all permutations of n items is the symmetric group denoted by $\mathfrak{S}_n$, which has the group operation of composition. Full orderings or permutations are denoted by listing the items according to their ranks, separated by vertical bars as in [20]. For example, for $n = 3$, one permutation ranking

[1]A popular discount function is the logarithm function [21] but other discount functions have been proposed as well.

item 2 as first and 1 as last is denoted as $\pi^{-1}(1)|\pi^{-1}(2)|\pi^{-1}(3) = 2|3|1$.

Complete with-ties rankings are similar to complete without-ties rankings but they allow some of the items to be of tied rank. It occurs when judges do not provide enough information to construct a total order. In particular, we define tied rankings as a partition of $\{1, \ldots, n\}$ to $k < n$ disjoint subsets $A_1, \ldots, A_k \subset \{1, \ldots, n\}$ such that all items in A_i are preferred to all items in A_{i+1} but no information is provided concerning the relative preference of the items among the sets A_i. We denote such rankings by separating the items in A_i and A_{i+1} with a $\prec$ or $|$ notation. For example, the tied ranking $A_1 = \{3\}, A_2 = \{2\}, A_3 = \{1, 4\}$ (items 1 and 4 are tied for last place) is denoted as $3 \prec 2 \prec 1, 4$ or $3|2|1, 4$.

Conceptually we take a tie to be a lack of information with the notion that more information could in principle breaks the tie. This means that a permutation with ties can be thought of as a set of permutations where each member of the set is the potential true permutation if we have the full information. This idea is incorporated in the term full ranking, which is denoted as a ranking without-ties. For example the with-ties ranking $3|2|1, 4$ corresponds to the following set of permutations:

$$3|2|1, 4 = \{3|2|1|4; 3|2|4|1\} \in \mathfrak{S}_4.$$

For a complete with-ties ranking of k compartments (partitions), each of size $|A_i|$, the size of the set of permutations corresponding to it is $\prod_{i=1}^{k} |A_i|!$.

Popular types of complete with-ties rankings include top-k rankings, such as the web pages output by search engines in response to a query where the top-k items are a full ordering and the rest $n - k$ items are tied as the bottom ranks. The size of the set of permutations corresponding to a top-k ranking is $(n - k)!$.

Incomplete rankings occur when judges omit certain items from their preference information altogether. For example assuming a set of items $\{1, \ldots, 4\}$, a judge may report a preference $3 \prec 2 \prec 4$, omitting altogether item 1 which the judge did not observe or experience. This case is very common in situations involving a large

number of items n. In this case judges typically provide preference only for the $l \ll n$ items that they have observed or experienced. For example, in movie recommendation systems we may have $n \sim 10^3$ and $l \sim 10^1$.

Incomplete rankings may also be described by the set of permutations that are consistent with it. For example,

$$3 \prec 2 \prec 4 = \{1 \prec 3 \prec 2 \prec 4\} \cup \{3 \prec 1 \prec 2 \prec 4\} \cup \{3 \prec 2 \prec 1 \prec 4\} \cup \{3 \prec 2 \prec 4 \prec 1\}$$

is a set of four permutations corresponding to the incomplete rankings.

Tied-incomplete rankings are the rankings with ties and missing items. A typical example is a list of movies with rated scores (0-5) provided by a user, where not all movies are rated and movies of the same score are tied.

2.3 Distances and Dissimilarities on Rankings

Popular distance measures between permutations are Kendall's tau, Spearman's rho, Footrule, Ulam's distance, Cayley's distance [20], and Hoeffding distance [65].

- Spearman and Footrule (d_p distance):

$$d_p(\pi, \sigma) = \sum_{r=1}^{n} |\pi(r) - \sigma(r)|^p \tag{1}$$

  for some $0 < p < \infty$. The Spearman and Footrule distances use $p = 2$ and $p = 1$ respectively.

- Hoeffding:

$$d_{hoeff}(\pi, \sigma) = \sum_{r=1}^{n} a(\pi(r), \sigma(r)), \tag{2}$$

  where a is a function on $\{1, \ldots, n\} \times \{1, \ldots, n\}$ that satisfies $a(i, i) = 0$ and $a(i, j) = a(j, i)$. They generalize d_p distances.

- Kendall:

$$d_{ken}(\pi, \sigma) = \sum_{i=1}^{n-1} \sum_{l>i} I(\pi\sigma^{-1}(i) - \pi\sigma^{-1}(l)), \tag{3}$$

where $I(x) = 1$ for $x > 0$ and 0 otherwise. Kendall's tau is the minimum number of transpositions of adjacent items needed to bring π to σ.

Instead of counting adjacent item swaps as in Kendall's, Carley's distance [20] is the minimum number of arbitrary pairwise interchanges needed to bring π to σ. Ulam's distance [20] is minimum number of removing and reinserting an item at places to bring π to σ.

There are several ways to extend these permutation distances to partial incomplete rankings including Hausdorff distance [20] and expected distances [3, 65]. Denote S and R as two sets corresponding to two partial incomplete rankings, which consists of all permutations that are consistent with the two observed rankings. The Hausdorff distance and the expected dissimilarity are defined below:

- Hausdorff

$$\rho_{hous}(S, R) = \max\{\max_{\pi \in S} \min_{\sigma \in R} d(\pi, \sigma), \max_{\sigma \in R} \min_{\pi \in S} d(\pi, \sigma)\}. \tag{4}$$

- Expected Dissimilarity

$$\rho(S, R) = \frac{1}{|S| \cdot |R|} \sum_{\pi \in S} \sum_{\sigma \in R} d(\pi, \sigma), \tag{5}$$

where the distance $d(\pi, \sigma)$ for permutations can be any of the aforementioned metrics. The expectation is calculated with respect to a uniform distribution over the sets of consistent rankings.

The resulted Hausdorff distance is a metric. A drawback to Hausdorff distance is it lacks simple closed form formulas. On the other hand, the expected dissimilarity is not a metric since the expected distance between S and itself is not zero i.e., $\rho(S, S) \neq 0$ if $|S| > 1$. However, for some distance metric $d(\pi, \sigma)$ and types of rankings, an efficient closed form formula exists for the expected dissimilarity. In this work, we mainly focus on the expected dissimilarities for partial incomplete rankings.

In Section, 2.3.1, we present the closed form formula of the expected dissimilarity using Kendall's tau on partial incomplete rankings.

2.3.1 Kendall's Tau Distance and Expectation Formulas

The expected dissimilarity $\rho(S, R)$ in Equation (5) for two partial incomplete rankings has a simple formula when adopting the Kendall's tau distance in (3).

Proposition 1. *For two sets of permutations S, R corresponding to tied-incomplete rankings*

$$\frac{1}{|S||R|}\sum_{\pi\in S}\sum_{\sigma\in R} d_{ken}(\pi,\sigma) = \frac{n(n-1)}{4} - \frac{1}{2}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(1-2p_{ij}(S))(1-2p_{ij}(R)) \qquad (6)$$

$$p_{ij}(U) = \begin{cases} 1/2 & i \text{ and } j \text{ are ranked in } U \text{ with } \tau_U(i) = \tau_U(j) \\ I(\tau_U(j) - \tau_U(i)) & i \text{ and } j \text{ are ranked in } U \text{ with } \tau_U(i) \neq \tau_U(j) \\ 1 - \frac{\tau_U(i) + \frac{\phi_U(i)-1}{2}}{k+1} & \text{only } i \text{ is ranked in } U \\ \frac{\tau_U(j) + \frac{\phi_U(j)-1}{2}}{k+1} & \text{only } j \text{ is ranked in } U \\ 1/2 & \text{otherwise} \end{cases}$$

with $\tau_U(i) = \min_{\pi\in U}\pi(i)$, and $\phi_U(i)$ being the number of items that are tied to i in U.

Proof. Note that (6) is an expectation with respect to the uniform measure. We thus start by computing the probability $p_{ij}(U)$ that i is preferred to j for $U = S$ and $U = R$ under the uniform measure. Five scenarios exist for each of $p_{ij}(U)$ corresponding to whether each of i and j are ranked by S, R. Starting with the case that i is not ranked j is ranked we note that i is equally likely to be preferred to any item preferred to and including j and that there are $k+1$ possible rankings for i. Given the uniform distribution over compatible rankings item j is equally likely to appear in positions $\tau_U(j), \ldots, \tau_U(j) + \phi_U(j) - 1$. Thus

$$p_{ij} = \frac{1}{\phi_U(j)}\frac{\tau_U(j)}{k+1} + \cdots + \frac{1}{\phi_U(j)}\frac{\tau_U(j)+\phi_U(j)-1}{k+1} = \frac{\tau_U(j) + \frac{\phi_U(j)-1}{2}}{k+1}.$$

Similarly, if j is unknown and i is known then $p_{ij}+p_{ji} = 1$. If both i and j are unknown either ordering must be equally likely given the uniform distribution making $p_{ij} = 1/2$. Finally, if both i and j are known $p_{ij} = 1, 1/2, 0$ depending on their preference. Given p_{ij}, linearity of expectation, and the independence between rankings, the change in the expected number of inversions relative to the uniform expectation $n(n-1)/4$ can be found by considering each pair separately,

$$\begin{aligned} \mathrm{E}d(i,j) &= \frac{1}{2}P\left(i \text{ and } j \text{ disagree}\right) - \frac{1}{2}P\left(i \text{ and } j \text{ agree}\right) \\ &= \frac{1}{2}(p_{ij}(\sigma)(1-p_{ij}(\pi)) + (1-p_{ij}(\sigma))p_{ij}(\pi)) \\ &\quad -p_{ij}(\sigma)p_{ij}(\pi) - (1-p_{ij}(\sigma))(1-p_{ij}(\pi))) \\ &= \frac{-1}{2}\left(1-2p_{ij}(\sigma)\right)\left(1-2p_{ij}(\pi)\right). \end{aligned}$$

Summing the $n(n-1)/2$ components yields the desired quantity. □

2.4 The Weighted Hoeffding Distance

The weighted Hoeffding distance has three main nice properties. First, the weighted Hoeffding distance is a metric. Second, the distance is customized to emphasize disagreements over top ranks by controlling the weight. When the weight is uniform with respect to ranks, the weighted Hoeffding distance is equivalent to Kendall'tau, Spearman's rho, or Footrule. Third, the expected weighted Hoeffding distance for top-k rankings maintains the second property and is computational efficient. The first and second properties are presented in Section 2.4.1 and the third property is presented in Section 2.4.2.

2.4.1 Definitions and Properties

The weighted Hoeffding distance $d_w(\pi, \sigma)$ is a variation of the earth mover's distance[2] [81] on permutations. It may also be regarded as a weighted version of the Hoeffding

[2]The earth mover's distance between two non-negative valued function is the minimum amount of work needed to transform one to the other, when the functions are viewed as representing spatial distributions of earth or rubble.

distance [65]. It is best described as the minimum amount of work needed to transform the permutation π to σ. Work, in this case, is the total amount of work needed to bring each item from its rank in π to its rank in σ i.e., the r-item is transported from rank $k = \pi(r)$ to $l = \sigma(r)$ (for all $r = 1, \ldots, n$) requiring $w_k + \cdots + w_{l-1}$ work (assuming $k < l$) where w_k is the work required to transport an item from rank k to $k + 1$. For example, the distance $d(1|2|3, 2|1|3)$ is $w_1 + w_1$ due to the sequence of moves $1|2|3 \to |1, 2|3 \to 2|1|3$. Another example is $d(1|2|3, 3|1|2) = w_1 + w_2 + w_2 + w_1$ due to the sequence of moves $1|2|3 \to |1, 2|3 \to |1|2, 3 \to |1, 3|2 \to 3|1|2$.

Formally, the distance may be written as

$$d_w(\pi, \sigma) = \sum_{r=1}^{n} d'_w(\pi(r), \sigma(r)) \quad \text{where} \tag{7}$$

$$d'_w(u, v) = \begin{cases} \sum_{t=u}^{v-1} w_t & \text{if } u < v \\ d'_w(v, u) & \text{if } u > v \\ 0 & \text{otherwise} \end{cases} . \tag{8}$$

The weight vector $w = (w_1, \ldots, w_{n-1})$ allows differentiating the work associated with moving items across top and bottom ranks. When the weight vector is constant, the weighted Hoeffding distance is equivalent to Kendall'tau, Spearman's rho or Footrule. A monotonic decreasing weight vector, e.g., $w_t = t^{-q}$, $t = 1, \ldots, n-1$, $q \geq 0$ correctly captures the fact that disagreements in top ranks should matter more than disagreements in bottom ranks [35, 48, 70]. The exponent q is the corresponding rate of decay. A linear or slower rate $0 \leq q \leq 1$ may be appropriate for persistent search engine users who are not very deterred by low-ranking websites. Choosing $q \to 0$ retrieves a weighting mathematically similar to the log function weighting that is used in NDCG [46] to emphasize top ranks. A quadratic or cubic decay $q = 2, 3$ may be appropriate for users who do not pay substantial attention to bottom ranks. The weight may be modified to $w_t = \max(t^{-q} - \epsilon, 0)$, $\epsilon > 0$ to capture the fact that many users simply do not look at results beyond a certain rank. While it is possible

to select an intuitive value of q, it is more desirable to select one that agrees with user studies.

Proposition 2. *Assuming w is a positive vector, the weighted Hoeffding distance* (7) *is a metric.*

Proof. Non-negativity $d_w(\pi, \sigma) \geq 0$ and symmetry $d_w(\pi, \sigma) = d_w(\sigma, \pi)$ are trivial to show. Similarly it is easy to see that $d_w(\pi, \sigma) = 0$ iff $\pi = \sigma$. The triangle inequality holds as

$$\begin{aligned} d_w(\pi, \sigma) + d_w(\sigma, \varphi) &= \sum_{r=1}^{n} d'_w(\pi(r), \sigma(r)) + d'_w(\sigma(r), \varphi(r)) \\ &\geq \sum_{r=1}^{n} d'_w(\pi(r), \varphi(r)) = d_w(\pi, \varphi), \end{aligned} \tag{9}$$

where the inequality (9) holds due to the positivity of w. □

2.4.2 Expectations for Top-k Rankings

In this section, we show that a closed form formula of the expected distance between top-k rankings using the weighted Hoeffding distance exists and is computational efficient.

A top-k ranking (e.g., a ranked list of websites output by a search algorithm) forms an ordered list $\langle i_1, \ldots, i_k \rangle$ of a subset of the items (e.g., websites) $\{i_1, \ldots, i_k\} \subset \{1, \ldots, n\}$. Different rankings (e.g., ranked lists output by search strategies) may result in lists of different sizes but in general k is much smaller than n. In addition to the notation $\langle i_1, \ldots, i_k \rangle$ we also denote it using the bar notation as

$$i_1|i_2|\cdots|i_k|i_{k+1}, \ldots, i_n \quad \text{where} \quad \{i_{k+1}, \ldots, i_n\} = \{1, \ldots, n\} \setminus \{i_1, i_2, \cdots, i_k\}, \tag{10}$$

indicating that the unranked items $\{1, \ldots, n\} \setminus \{i_1, \ldots, i_k\}$ are ranked after the k items. The top-k ranking (10) is a type of partial rankings, as there is no known preference among the unranked items $\{1, \ldots, n\} \setminus \{i_1, \ldots, i_k\}$. We therefore omit vertical lines between these items and list them separated by commas, i.e., $3|2|1, 4$ is

equivalent to the rankings $\langle 3, 2\rangle$, which prefers 3 to 2 and ranks 1 and 4 last without clear preference between them.

It is natural to identify a ranking $\langle i_1, \ldots, i_k\rangle$ as a full permutation of the items (e.g., websites) that is unknown except for the fact that it agrees with the item ranking in $\langle i_1, \ldots, i_k\rangle$. Denoting the set of permutations whose item ordering does not contradict $\langle i_1, \ldots, i_k\rangle$ as $\mathfrak{S}(\langle i_1, \ldots, i_k\rangle)$, we have that $\langle i_1, \ldots, i_k\rangle$ corresponds to a random draw from $\mathfrak{S}(\langle i_1, \ldots, i_k\rangle)$. Assuming lack of additional knowledge, we consider all permutations in $\mathfrak{S}(\langle i_1, \ldots, i_k\rangle)$ as equally likely resulting in

$$\begin{aligned}\rho(\langle i_1, \ldots, i_k\rangle, \langle j_1, \ldots, j_l\rangle) &= \mathsf{E}_{\pi\sim\mathfrak{S}(\langle i_1,\ldots,i_k\rangle),\sigma\sim\mathfrak{S}(\langle j_1,\ldots,j_l\rangle)} d(\pi, \sigma) \\ &= \frac{1}{(n-k)!(n-l)!} \sum_{\pi\in\mathfrak{S}(\langle i_1,\ldots,i_k\rangle)} \sum_{\sigma\in\mathfrak{S}(\langle j_1,\ldots,j_l\rangle)} d(\pi, \sigma). \quad (11)\end{aligned}$$

For example, consider the case of $n = 5$ with two rankings (e.g., the output of search strategies): $\langle 3, 1, 4\rangle = 3|1|4|2, 5$ and $\langle 1, 5\rangle = 1|5|2, 3, 4$. The expected distance is

$$\begin{aligned}\rho(3|1|4|2, 5, 1|5|2, 3, 4) = \frac{1}{2\cdot 6}(&d(3|1|4|2|5, 1|5|2|3|4) + d(3|1|4|5|2, 1|5|2|3|4) \\ &+ \cdots + d(3|1|4|2|5, 1|5|4|3|2) + d(3|1|4|5|2, 1|5|4|3|2)). \quad (12)\end{aligned}$$

Expression (11) provides a natural mechanism to incorporate information from partial rankings. It is difficult to compare directly two rankings $\langle i_1, \ldots, i_k\rangle$, $\langle j_1, \ldots, j_l\rangle$ of different sizes. However, the permutations in $\mathfrak{S}(\langle i_1, \ldots, i_k\rangle)$ and $\mathfrak{S}(\langle j_1, \ldots, j_k\rangle)$ are directly comparable to each other as they are permutations over the same set of websites. The expectation (11) aggregates information over such directly comparable events to provide a single interpretable and coherent dissimilarity measure.

The expectation defining ρ in (11) appears to require insurmountable computation as it includes summations over $(n-k)!(n-l)!$ elements with n being the size of the web. However, using techniques similar to the ones developed in [65] we are able to derive the following closed form.

Proposition 3. *The following closed form applies to the expected distance over the weighted Hoeffding distance* (7)*.*

$$\rho(\langle i_1,\ldots,i_k\rangle,\langle j_1,\ldots,j_l\rangle)=\sum_{r=1}^{n}\bar{d}(r)\quad \textit{where} \tag{13}$$

$$\bar{d}(r)=\begin{cases} d'_w(u,v) & r\in A\cap B \\ \frac{1}{n-l}\sum_{t=l+1}^{n} d'_w(t,u) & r\in A\cap B^c \\ \frac{1}{n-k}\sum_{t=k+1}^{n} d'_w(t,v) & r\in A^c\cap B \\ \frac{1}{n-k}\frac{1}{n-l}\sum_{t=k+1}^{n}\sum_{s=l+1}^{n} d'_w(t,s) & \textit{otherwise} \end{cases}.$$

Above, $A=\{i_1,\ldots,i_k\}$*,* $B=\{j_1,\ldots,j_l\}$*, and* $u\in\{1,\ldots,k\}$*,* $v\in\{1,\ldots,l\}$ *are the respective ranks of* r *in* $\{i_1,\ldots,i_k\}$ *and* $\{j_1,\ldots,j_l\}$ *(if they exist).*

Proof. A careful examination of (7) reveals that it may be written in matrix notation:

$$d(\pi,\sigma)=\mathrm{tr}(A_\pi\triangle A_\sigma^T), \tag{14}$$

where $\mathrm{tr}(\cdot)$ is the trace operator, $\triangle$ is the $n\times n$ distance matrix with elements $\triangle_{uv}=d'_w(u,v)$, and A_π, A_σ are permutation matrices corresponding to the permutations π and σ i.e., $[A_\pi]_{uv}=1$ iff $\pi(u)=v$. Using Equation (14), we have

$$\begin{aligned}\rho(\langle i_1,\ldots,i_k\rangle,\langle j_1,\ldots,j_l\rangle)&=\frac{\sum_{\pi\in\mathfrak{S}(\langle i_1,\ldots,i_k\rangle)}\sum_{\sigma\in\mathfrak{S}(\langle j_1,\ldots,j_k\rangle)}\mathrm{tr}(A_\pi\triangle A_\sigma^T)}{(n-k)!(n-l)!}\\&=\mathrm{tr}(\hat{M}_{\langle i\rangle}\triangle(\hat{M}_{\langle j\rangle})^T),\end{aligned} \tag{15}$$

where

$$\hat{M}_{\langle i\rangle}=\frac{\sum_{\pi\in\mathfrak{S}(\langle i_1,\ldots,i_k\rangle)}A_\pi}{(n-k)!},\quad \hat{M}_{\langle j\rangle}=\frac{\sum_{\sigma\in\mathfrak{S}(\langle j_1,\ldots,j_l\rangle)}A_\sigma}{(n-l)!}. \tag{16}$$

Note that the marginal matrices $\hat{M}_{\langle i\rangle}$, $\hat{M}_{\langle j\rangle}$ have a probabilistic interpretation as their u,v entries represent the probability that item u is ranked at v. Combining (15) with Lemma 1 below completes the proof. □

Lemma 1. *Let $\hat{M}$ be the marginal matrix for a top-k rankings $\langle i_1, \ldots, i_k \rangle$ with a total of n items as in* (16)*. If $r \in \{i_1, \ldots, i_k\}$ and $r = i_s$ for some $s = 1, \ldots, k$, then $\hat{M}_{rj} = \delta_{js}$ where $\delta_{ab} = 1$ if $a = b$ and 0 otherwise. If $r \notin \{i_1, \ldots, i_k\}$ then $\hat{M}_{rj} = 0$ for $j = 1, \ldots, k$ and $1/(n-k)$ otherwise.*

Proof. For a top-k ranking $\langle i_1, \ldots, i_k \rangle$ out of n items, the size of the set $\mathfrak{S}(\langle i_1, \ldots, i_k \rangle)$ is $(n-k)!$. Each of the permutations compatible with it has exactly the same top-k ranks. If $r \in \{i_1, \ldots, i_k\}$ and $r = i_s$ for some $s = 1, \ldots, k$ then the number of permutations compatible with $\langle i_1, \ldots, i_k \rangle$ that assign rank s to the item is $(n-k)!$. Similarly, the number of consistent permutations assigning rank other than s to the item is 0. As a result we have $\hat{M}_{rs} = \frac{(n-k)!}{(n-k)!} = 1$ and $\hat{M}_{rj} = 0$ for $j \neq s$. If $r \notin \{i_1, \ldots, i_k\}$, the number of permutations consistent with the rankings that assign rank $j \in \{k+1, \ldots, n\}$ to the item is $(n-k-1)!$. Similarly, the number of permutations that assign rank $j \in \{1, \ldots, k\}$ to the item is 0. As a result $\hat{M}_{rj} = 0$ for $j = 1, \ldots, k$, and $\hat{M}_{rj} = \frac{(n-k-1)!}{(n-k)!} = \frac{1}{n-k}$ for $j = k+1, \ldots, n$. □

The expected distance (11) may be computed very efficiently, assuming that some combinatorial numbers are pre-computed offline. Bounding k, l by a certain number $k, l \leq m \ll n$ we have that the online complexity is $O(k+l)$ and the offline complexity is $O(n + m^2)$. The next proposition makes this precise. A pseudo-code description of the distance computation algorithm is given as Algorithm 1.

Proposition 4. *Let $\langle i_1, \ldots, i_k \rangle$ and $\langle j_1, \ldots, j_l \rangle$ be top-k and top-l ranks on a total n items with $k, l \leq m \ll n$. Assuming that $d_w(u, v)$ is computable in constant time and space complexity (as is the case for many polynomial decaying weight vectors w) the online space and time complexity is $O(k + l)$. The offline space complexity is $O(m^2)$ and the offline time complexity is $O(n + m^2)$.*

Proof. From Equation (13), the offline pre-computation requires computing $D_{m \times m}$, $D^{E1}_{m \times m}$ and $D^{E2}_{m \times m}$, where $D_{uv} = d'_w(u, v)$, $D^{E1}_{kv} = \frac{1}{n-k} \sum_{t=k+1}^{n} d'_w(t, v)$, and $D^{E2}_{kl} =$

$\frac{1}{n-k}\frac{1}{n-l}\sum_{t=k+1}^{n}\sum_{s=l+1}^{n} d'_w(t,s)$. The space complexity for computing these matrices is $O(m^2)$. The time complexity to compute $D_{m\times m}$ is $O(m^2)$. Exploiting features of cumulative sums and the matrix $D_{m\times m}$ it can be shown that computing $D^{E1}_{m\times m}$ requires $O(n+m^2)$ time. Similarly, computing $D^{E2}_{m\times m}$ requires $O(n+m^2)$ time. As a result, the total offline complexity is $O(m^2)$ space and $O(n+m^2)$ time. Given the three precomputed matrices, computing the expected distance for two partial rankings $\langle i_1,\ldots,i_k\rangle$ and $\langle j_1,\ldots,j_l\rangle$ requires $O(k+l)$ time and space. The reasons are that given two lists, the time to identify overlapping items from two lists of size k and l is $O(k+l)$ and that for items ranked by at least one engine, we need to use the look-up table no more than $k+l$ times and another extra look-up for items never ranked in both. □

2.4.3 Experiments and Results

The weighted Hoeffding distance has several nice properties that make it more appropriate for measuring dissimilarities between top-k rankings than other permutation measures. First, it is a true metric in contrast to the generalized Kendall's tau [27]. Second, it allows customization to different users who pay varying degrees of attention to items (websites) in different ranks (typically higher attention is paid to higher ranks). Standard permutation distances such as Kendall's tau, Spearman's rho, Footrule, Ulam's distance, and Cayley's distance treat all ranks uniformly [23]. Its clear interpretation allows explicit specification of the weight vector based on user studies. Finally, it is computationally tractable to compute the weighted Hoeffding distance as well as its expectation over partial rankings corresponding to ρ in (11).

The expected weighted Hoeffding distance ρ can be extended to search algorithm dissimilarity $\overline{\rho}$ in Equation (19) by taking another expectation over a set of queries. We list five desired properties of distance measures for search algorithms as the following: (i) symmetric, (ii) interpretable with respect to search algorithms retrieving

Algorithm 1 Algorithm to compute expected weighted Hoeffding distance between two partial incomplete rankings π and σ. The online complexity of the algorithm is $O(k_1 k_2)$. A slightly more complex algorithm can achieve online complexity $O(k_1 + k_2)$ as described in Proposition 4.

Off-line:

1. Specify n, the number of total items and m the list length bound.
2. Precompute matrices $D_{m\times m}$, $D^{E1}_{m\times m}$ and $D^{E2}_{m\times m}$ (Proposition 4).

On-line:

3. Call Expected-Weighted-Hoeffding(π, σ) for lists π and σ.

function Expected-Weighted-Hoeffding(π,σ)
$k_1 = size(\pi)$
$k_2 = size(\sigma)$
$[\pi_{mark}, \sigma_{mark}] =$ Mark-Rank(π, σ)
$sum = 0$
for $i = 1 \rightarrow k_1$ **do**
  if $\pi_{mark}[i] > 0$ **then**
    $sum = sum + D[i, \pi_{mark}[i]]$
  else
    $sum = sum + D^{E1}[k_2, i]$
  end if
end for
$count = 0$
for $i = 1 \rightarrow k_2$ **do**
  if $\sigma_{mark}[i] == 0$ **then**
    $sum = sum + D^{E1}[k_1, i]$
    $count = count + 1$
  end if
end for
$sum = sum + (n - k_1 - count) \cdot D^{E2}[k_1, k_2]$
return sum

function Mark-Rank(a,b)
$k_1 = size(a)$
$k_2 = size(b)$
$a_{mark} = zeros(1 \ldots k_1), b_{mark} = zeros(1 \ldots k_2)$
for $i = 1 \rightarrow k_1$ **do**
  for $j = 1 \rightarrow k_2$ **do**
    if $a[i] = b[j]$ **then**
      $a_{mark}[i] = j$
      $b_{mark}[j] = i$
    end if
  end for
end for
return $[a_{mark}, b_{mark}]$

ranked lists of different lengths, (iii) flexible enough to model the increased attention users pay to top ranks over bottom ranks, (iv) computationally efficient, and (v) aggregate information over multiple queries in a meaningful way. Table 1 summarizes the advantages of the expected weighted Hoeffding distance over other dissimilarities.

Table 1: Summary of how different dissimilarities satisfy properties (i)-(v).

	(i)	(ii)	(iii)	(iv)	(v)
Kendall/Spear [23]	✓			✓	
Fligner Kendall [27]			✓		
E Kendall top k [26]	✓	✓		✓	✓
E Spearman [62]	✓	✓		✓	✓
InverseMeasure [6]	✓		✓	✓	✓
NDCG [46]			✓	✓	✓
E Weighted Hoeffding	✓	✓	✓	✓	✓

We also conduct some comparisons between the weighted Hoeffding distance and alternative distance measures. Table 2 shows how one recently proposed measure, the inverse measure [6], lacks discriminative power, assigning the same dissimilarity to very different rankings. Kendall's tau and the other distances proposed in [23, 26] lack the ability to distinguish disagreement in top ranks and bottom ranks. In particular, Kendall's tau is identical to our weighted Hoeffding distance with uniform weights (see Figures 4-5 in Chapter 3 for a demonstration of its inadequacy). NDCG [46] and other precision recall measures rely on comparing a ranked list to a ground truth of relevant and not-relevant documents or websites. As such they are not symmetric and are inappropriate for comparing partial incomplete rankings.

Table 3 shows the changes of weighted Hoeffding distances with weight $w_t = t^{-3}$ as the number of total items n changes. It reveals that increasing n beyond a certain size does not alter the distances between partial incomplete rankings. This indicates a lack of sensitivity to the precise value of n as well as computational speedup resulting from replacing n by $n' \ll n$.

Table 2: A comparison of the inverse measure [6] with the weighted Hoeffding distance indicates that the inverse measure lacks discriminative power as it assigns the same dissimilarity to very different rankings ($n = 5$).

d to $1\|2\|3\|4\|5$	InverseMeasure	$w_t = t^{-1}$	$w_t = t^{-2}$
2	0.8374	0.6500	0.7539
3	0.8374	0.7786	0.8589
4	0.8374	0.8357	0.8901
5	0.8374	0.8571	0.8988
$1\|3$	0.2481	0.3048	0.2049
$1\|4$	0.2481	0.3810	0.2464
$1\|5$	0.2481	0.4095	0.2581

Table 3: A comparison of weighted Hoeffding distance with cubic weight decay $w_t = t^{-3}$ reveals that increasing n beyond a certain size does not alter the distances between partial incomplete rankings. This indicates lack of sensitivity to the precise value of n as well as computational speedup resulting from replacing n by $n' \ll n$.

d to $1\|2\|3\|4\|5$	$n = 5$	$n = 10$	$n = 10^3$	$n = 10^5$	$n = 10^7$
$1\|2\|3\|5\|4\|$	0.0117	0.0176	0.0670	0.0698	0.0699
$2\|1\|3\|4\|5\|$	0.7464	0.6755	0.6660	0.6683	0.6683
$1\|4\|2\|$	0.1268	0.1362	0.1950	0.1980	0.1981
$1\|$	0.1064	0.1592	0.2656	0.2692	0.2692
$2\|1$	0.7726	0.7283	0.7515	0.7543	0.7543
$5\|$	0.9395	0.9280	0.9820	0.9851	0.9852
$5\|4\|3\|2\|1\|$	1.0000	0.9025	0.8727	0.8748	0.8748

CHAPTER III

VISUALIZING PARTIAL RANKINGS

In this chapter, we present a computational efficient framework for visualizing top-k ranking data for large n. The framework works by projecting rankings into a low dimensional continuous vector space using multi-dimensional scaling (MDS) based on the dissimilarity described in Chapter 2. Shepard's plot can be used to validate the MDS embedding. To evaluate the proposed framework we conduct three sets of analyses. The first set of analyses (in Section 3.3) uses synthetic data to examine properties of the embedding. In particular, the framework is applied to visualizing simulated top-k rankings using different distance metrics. The embeddings indicate that, as described in Chapter 2, the weighted Hoeffding distance is more appropriate than the other traditional distance metrics such as Kendall's tau for top-k rankings. The second set of experiments (in Section 3.4) includes real world search engine data, the goal of which is to demonstrate the benefit and applicability of the framework in the context of a real world visualization problem. The third set of experiments (in Section 3.5) visualize ranking diversity for single or multiple search engines, given a set of queries representing the most common reformulations (intents).

3.1 Related Work

Visualization for partial rankings on medium or large size of items is quite challenging. Popular visualization tool for permutations includes polytope [93] which is a convex hull in $\mathbb{R}^n$ space with $n!$ vertices each corresponding to a permutation. Two vertices are connected by an edge if the Kendall's tau distance is 1, which means the permutations differ only by transposing two adjacent items. However, for rankings on medium or large size of items, the use of the polytope for visualization purposes

[93, 5] is rather limited.

Common approaches for visualizing high dimensional ranking data is reducing the data on a lower dimensional space. Projections of polytope [31, 16] is a natural extension, which is similar to principle component analysis (PCA) or multidimensional scaling (MDS). The goal is find the projections that maximize the distance between the permutations and the center of the polytope. Other efforts including [95] map the rankings to a vector space and reduce it to 2D using stand dimensional reduction tools like PCA or MDS. However, the method induces distortions when transforming from ranking space to a high dimensional vector space.

While some previous work on visualizing ranked lists is based on item similarity, e.g., the top-k ranking visualization [80], there is renewed interest in visualizing and analyzing set-level differences between ranked lists. Fagin et al. [26] and Bar-Ilan et al. [6] investigate the relationship among ranked lists of search engines by examining the pairwise distance matrix but do not make the connection with visualization. Liggett and Buckley [62] use multi-dimensional scaling over ranking dissimilarity to examine search system variations due to the effect of query expansion, where the dissimilarity was based on Spearman's coefficient. In addition, tools like MetaCrystal [88] and the more general ConSet [52] regard ranked lists as a set of items and visualize the common items among them.

3.2 Multi-dimensional Scaling and Validation of Embeddings

There are several techniques for visualizing complex data such as rankings. One of the most popular techniques is multidimensional scaling (MDS) [19, 8], which transforms complex high dimensional data $s_1, \ldots, s_m$ into 2-D vectors $z_1, \ldots, z_m$ that are easily visualized by displaying them on a 2-D scatter plot. Assuming that a suitable dissimilarity measure between the high dimensional data $\rho^*(s_i, s_j)$ has been identified, MDS computes the 2-D embedding of the high dimensional data $s_i \mapsto z_i \in$

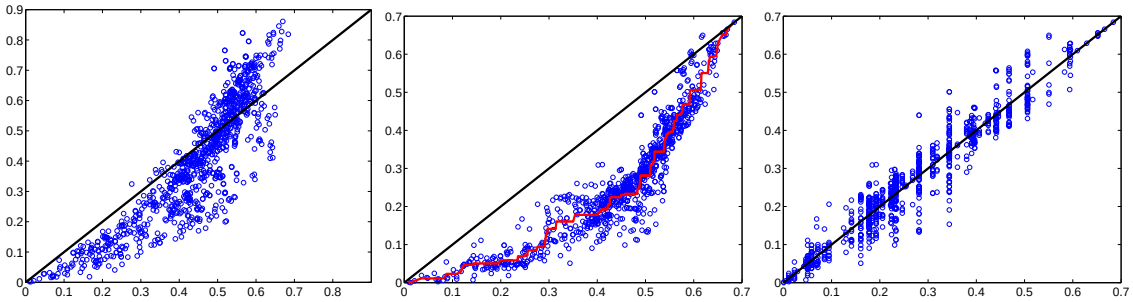


Figure 3: Shepard plots (embedding distances as a function of the original dissimilarities) for a 2D MDS embeddings. The metric stress MDS [8] (left panel) produces an embedding that has a lower overall distortion than the non-metric stress MDS [8] (middle panel). The non-metric stress MDS, however, achieves an embedding by transforming the original dissimilarities into alternative quantities called disparities using a monotonic increasing mapping (red line in middle panel). Doing so preserves the relative ordering of the distances thus accurately reflecting the spatial relationships between the points. The right panel displays the embedded distances as a function of the disparities. The displayed spread is symmetric and with little outliers making the non-metric MDS a viable alternative to the metric MDS. See [8] for more details.

$\mathbb{R}^2$, $i = 1, \ldots, m$ that minimizes the spatial distortion caused by the embedding

$$R(z_1, \ldots, z_m) = \sum_{i<j} \left(\rho^*(s_i, s_j) - \|z_i - z_j\|\right)^2. \tag{17}$$

In other words, the coordinates $z_1, \ldots, z_m$ in $\mathbb{R}^2$ corresponding to $s_1, \ldots, s_m$ are selected to minimize the distortion

$$(z_1, \ldots, z_m) = \underset{z'_1, \ldots, z'_m}{\arg\min}\, R(z'_1, \ldots, z'_m). \tag{18}$$

Variations of MDS with slightly different distortions (17) and objective functions (18) may be found in [8].

A central assumption of this framework is that MDS embeddings can give a faithful representation of the distances in the original higher-dimensional space. Thus, we now provide a validation of the MDS embedding as a visualization tool, diagnosing whether MDS is providing a reasonable embedding and which MDS variant should be preferable.

The most common tool for validating the MDS embedding is Shepard's plot (Figure 3, left) which displays a scatter plot contrasting the original dissimilarities (on the

x axis) and the corresponding distances after the embedding (on the y axis). Points on the diagonal represent zero distortion and a curve that deviates substantially from the diagonal represents substantial distortion. The Shepard's plot in Figure 3 (left) corresponds to the metric MDS using the standard stress criterion as described in (18). The plot displays low distortion with a tendency to undervalue dissimilarities in the range $[0, 0.5]$ and to overvalue dissimilarities in the range $[0.5, 0.9]$. Such a systematic discrepancy between the way small and large distances are captured is undesirable.

An alternative is non-metric MDS which achieves an embedding by transforming the original dissimilarities into alternative quantities called disparities using a monotonic increasing mapping which are then approximated by the embedding distances [8]. Doing so preserves the relative ordering of the original dissimilarities and thus (assuming the embedding distances approximate well the disparities) accurately represent the spatial relationship between the points. Figure 3 (middle) displays the Shepard's plot for the same data embedded using non-metric stress MDS with the disparities displayed as a red line. Figure 3 (right) displays the embedded distances as a function of the disparities revealing no systematic tendency to overestimate or underestimate as did the metric MDS. Thus, despite the fact that its numeric distortion is higher, the non-metric MDS is a viable alternative to the metric MDS, and is what was used to generate the figures in this work.

3.3 Simulation Study

In the simulation study, the framework is applied to visualizing simulated top-k rankings using different distance metrics. A typical example of top-k rankings is the ranked list returned by a search algorithm. The dissimilarity ρ^* in Equation (17) used to generate MDS embeddings is chosen from the expected weighted Hoeffding distance

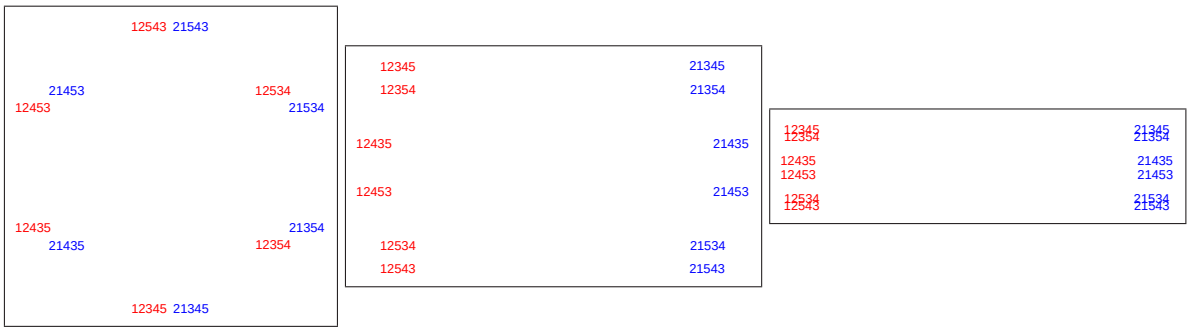


Figure 4: MDS embedding of permutations over $n = 5$ items (e.g., websites). The embeddings were computed using the weighted Hoeffding distance (7) with uniform weight function $w_t = 1$ (left), linear weight function $w_t = 1/t$ (middle) and quadratic weight function $w_t = 1/t^2$ (right). The permutations starting with 1 and 2 (colored in red) and the permutations starting with 2 and 1 (colored in blue) become more spatially disparate as the rate of weight decay increases. This represents the increased importance assigned to agreement in top ranks as we move from uniform to linear and quadratic decay.

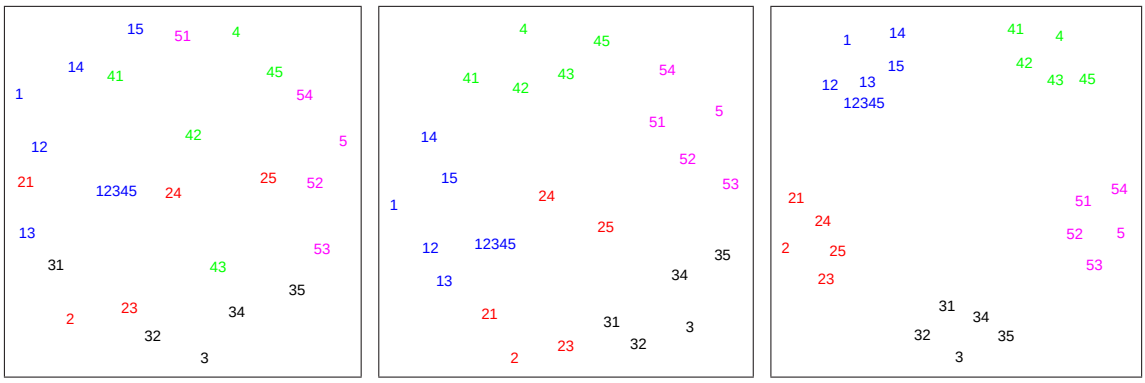


Figure 5: MDS embedding of ranked lists of varying lengths (k varies) over a total of $n = 5$ items (e.g., websites). The embeddings were computed using the expected weighted Hoeffding's distance (13) with uniform weight function $w_t = 1$ (left), linear weight function $w_t = 1/t$ (middle) and quadratic weight function $w_t = 1/t^2$ (right). We observe the same phenomenon that we saw in Figure 4 for permutations. The expected distance (13) separates ranked lists agreeing in their top ranks (denoted by different colors) better as the weights decay faster.

in Equation (13) and the expected Kendall's tau distance in Equation (6). The embeddings indicate that, as described in Chapter 2, the weighted Hoeffding distance is more appropriate than the other traditional distance metrics such as Kendall's tau for top-k rankings.

We start by examining the embedding of permutations over $n = 5$ items (e.g., websites). A small number is chosen intentionally for illustrative purposes. We consider two sets of permutations. The first set contains all permutations ranking item 1 first and item 2 second. The second set contains all permutations ranking item 2 first and item 1 second. Figure 4 displays the MDS embedding of these two sets of permutations based on the weighted Hoeffding's distance (see Section 2.4) with constant weights $w_t = 1$ (left), linear weight $w_t = t^{-1}$ (middle) and quadratic weight $w_t = t^{-2}$ (right). The first set of permutations is colored red and the second set blue.

The uniform weight MDS embedding does not pay particular attention to differing items in the top ranks and so the red and blue permutations are interspersed together. This is also the embedding obtained by using the Kendall's tau distance (see Section 2.3.1) as in [23, 26, 51]. Moving to linearly decaying and quadratic decaying weights increases the separation between these two groups dramatically. The differences in items occupying top ranks are emphasized while differences in items occupying bottom ranks are de-emphasized. This demonstrates the ineffectiveness of using Kendall's tau distance or uniform weight Hoeffding distance in the context of search engines, which are top-k rankings. The precise form of the weight - linear, quadratic, or higher decay rate depends on the degree to which a user pays more attention to higher ranked items (e.g., websites).

The second simulated experiment is similar to the first, but it contains partial rankings as opposed to permutations. We form five groups - each one containing partially ranked lists ranking a particular website at the top. Figure 5 displays the MDS embedding of these five sets of permutations based on the expected weighted

Hoeffding distance $\rho(\langle i_1, \ldots, i_k \rangle, \langle j_1, \ldots, j_l \rangle)$ in (13) using constant weights $w_t = 1$ (left), linear weights $w_t = t^{-1}$ (middle), and quadratic weights $w_t = t^{-2}$ (right). Ranked lists in each of the different groups are displayed in different colors.

We obtain a similar conclusion with the expected distance over partial rankings as we did with the distances over permutations. The five groups are relatively interspersed for uniform weights and get increasingly separated as the rate of weight decay increases. This indicates the fact that as the decay rate increases, disagreements in top ranks are emphasized over disagreements at bottom ranks.

3.4 Application: Visualizing Search Engine Algorithms

Search engines return ranked lists of documents or websites in response to a query, with the precise forms of the top-k rankings. The term *search algorithm* is an intentionally vague term corresponding to a mechanism for producing ranked lists of websites in response to queries. We focus on the following three interpretations of search algorithms: (a) different search engines, (b) a single search engine subject to different query manipulation techniques by the user, and (c) a single engine queried across different internal states.

We propose to visualize search algorithms in 2D embeddings via multi-dimensional scaling based on the expected weighted Hoeffding distance, which is effective and computational efficient. In particular, considering search algorithms s_i, s_j as functions from queries to ranked lists and denoting $s_i(q)$ as the partial rankings retrieved by s_i in response to the query q, the expected weighted Hoeffding distance for partial rankings $\rho(s_i(q), s_j(q))$ in Equation (11) can be extended to search algorithm dissimilarity $\overline{\rho}$ by taking the expectation with respect to queries q, which is sampled from a representative set of queries Q.

Formally, the dissimilarity $\overline{\rho}$ between search algorithms s_i and s_j used in MDS in

Equation (17) is defined as:

$$\overline{\rho}(s_i, s_j) = \mathsf{E}_{q\sim Q}\{\rho(s_i(q), s_j(q))\} = \mathsf{E}_{q\sim Q}\ \mathsf{E}_{\pi\sim\mathfrak{S}(s_i(q))}\mathsf{E}_{\sigma\sim\mathfrak{S}(s_j(q))}\{d_w(\pi,\sigma)\}, \qquad (19)$$

where $d_w(\pi,\sigma)$ is a weighted distance between permutations π,σ defined in Section 2.4, $\mathsf{E}_{\pi\sim\mathfrak{S}(s_i(q))}\mathsf{E}_{\sigma\sim\mathfrak{S}(s_j(q))}$ is the expectation with respect to permutations π,σ that are sampled from the sets of all permutations consistent with the ranked lists output by the two algorithms $s_i(q), s_j(q)$ (respectively), and $\mathsf{E}_{q\sim Q}$ is an expectation with respect to all queries sampled from a representative set of queries Q. In the absence of any evidence to the contrary, we assume a uniform distribution over the set of queries Q and over the sets of permutations consistent with $s_i(q), s_j(q)$. The expectation over q ensures that the last property in Table 1 is satisfied: aggregate information over multiple queries in a meaningful way.

We demonstrate the visualization framework using a collection of popular search engines, a representative set of queries, and frequently used query manipulation methods. Such visualization are highly effective at summarizing and analyzing insights on which search engines to use (see Section 3.4.1), what search strategies users can employ (see Section 3.4.2), and how search results evolve over time (see Section 3.4.3).

3.4.1 Search Engines Similarities

In the first experiment we visualize the similarities and differences between nine different search engines: `altavista.com`, `alltheweb.com`, `ask.com`, `google.com`, `lycos.com`, `live.com`, `yahoo.com`, `aol.com`, and `dogpile.com`. We collected 50 popular queries online in each of six different categories: company names, questions[1], sports, tourism[2], university names, and celebrity names. These queries form a representative sample of queries Q within each category over which we average the expected distance ρ according to (19). Table 4 shows several queries for each one of the topic

[1] `queries from http://answers.yahoo.com`
[2] `queries from http://en.wikipedia.org/wiki/Tourism`

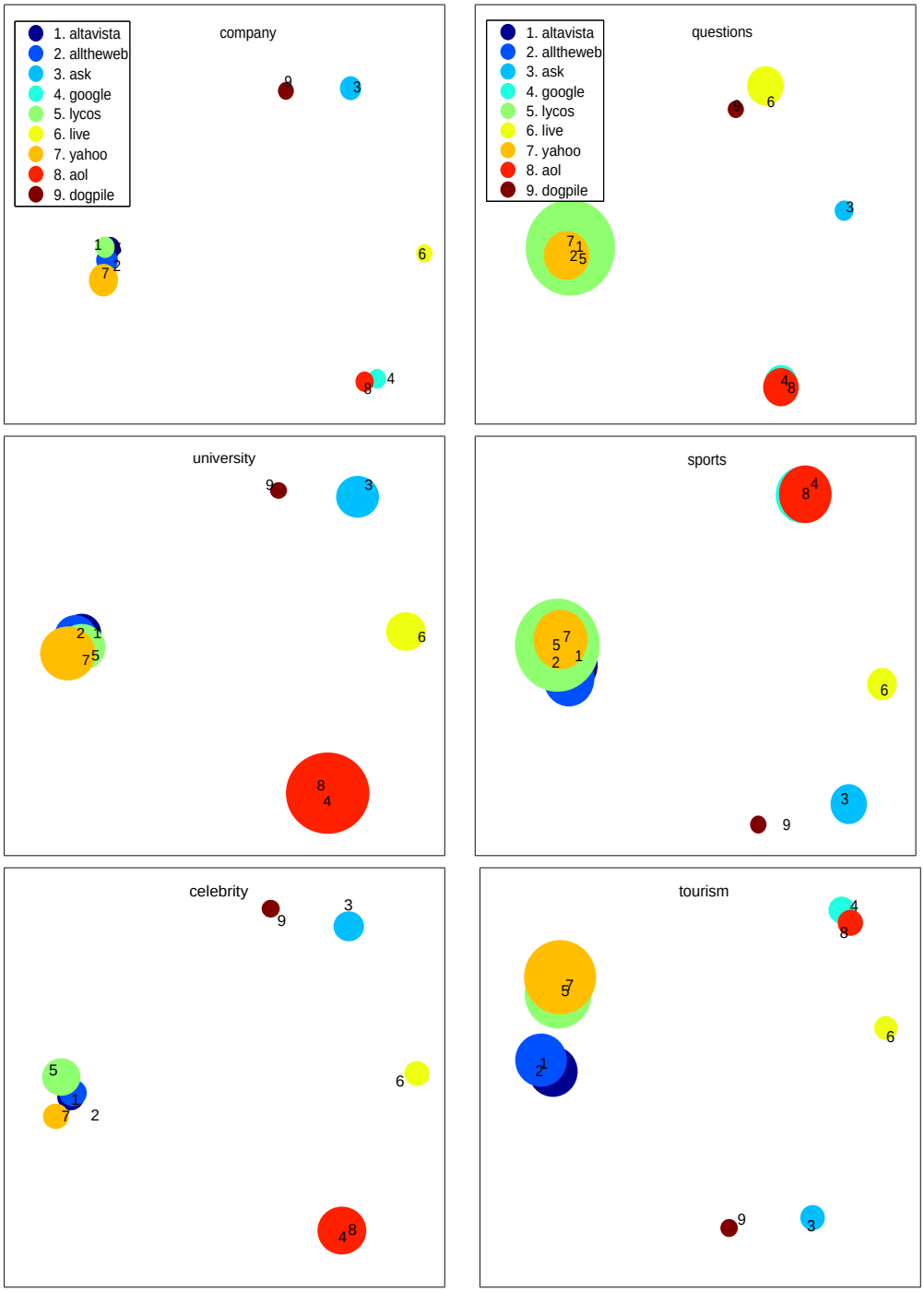


Figure 6: MDS embedding of search engine results over 6 sets of representative queries: company names, university names, celebrity names, questions, sports, and tourism. The MDS was based on the expected weighted Hoeffding distance with linear weighting $w_t = t^{-1}$ over the top 100 sites. Circle sizes indicate position variance with respect to within category queries.

Table 4: Selected queries from each of the 6 query categories, and from the set used for examining temporal variations.

Categories	Queries
Tourism	Times Square, Sydney Opera House, Eiffel Tower, Niagara Falls, Disneyland, British Museum, Giza Pyramids
Celebrity Names	Michael Bolton, Michael Jackson, Jackie Chan, Harrison Ford, Halle Berry, Whoopi Goldberg, Robert Zemeckis
Sports	Football, Acrobatics, Karate, Pole Vault, Butterfly Stroke, Scuba Diving,Table Tennis, Beach Volleyball, Marathon
University Names	Georgia Institute of Technology, University of Florida, Virginia Tech, University of California Berkeley
Company	Goldman Sachs, Facebook, Honda, Cisco Systems, Nordstrom, CarMax, Wallmart, American Express, Microsoft
Questions	How are flying buttresses constructed, Does toothpaste expire, How are winners selected for the Nobel Prize
Temporal Queries	AIG Bonuses, G20 major economies, Timothy Geitner, Immigration Policy, NCAA Tournament Schedule

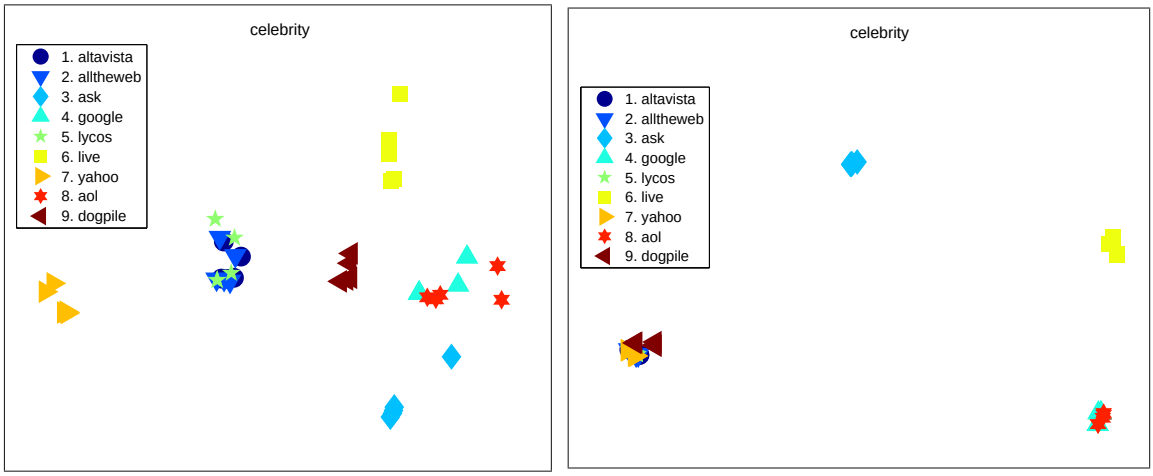


Figure 7: MDS embedding of search engine results over the query category celebrity with different query manipulations. The MDS was computed based on the expected distance (19) with (11) corresponding to the weighted Hoeffding distance with quadratic decaying weights (left panel) and Kendall top k distance [26] (right panel). Each marker represents a combination of one of the 9 search engines and one of the 5 query manipulation techniques. By comparison, the embedding of the Kendall top-k distance (right) lacks discriminative power and results in a loss of information.

categories. We visualize search result sets within each of the query categories in order to examine whether the discovered similarity patterns are specific to a query category, or are generalizable across many different query types.

Figure 6 displays the MDS embedding of each of the nine engines for the six query categories, based on the expected weighted Hoeffding distance (19) with linear weight decay. The ρ quantity is averaged over the 50 representative queries from that category. Each search engine is represented as a circle whose center is the 2-D coordinates obtained from the MDS embedding.

The radii of the circles in Figure 6 are scaled proportionally to the *positional stability* of the search engine. More precisely, we scale the radius of the circle corresponding to the i-th search engine proportionally to its distance variance over the 50 queries

$$\text{stability}(i) \stackrel{\text{def}}{=} \sum_{j:j\neq i} \mathsf{Var}_{q\sim Q}\{\rho(s_i(q), s_j(q))\}. \tag{20}$$

Scaling the circles according to (20) provides a visual indication of how much the position will change if one or more queries are deleted or added to Q. This can also be interpreted as the degree of uncertainty regarding the precise location of the search engines due to the averaging over Q.

Examining Figure 6 reveals several interesting facts. To begin with, there are five distinct clusters. The first and largest one contains the engines Altavista, Alltheweb, Lycos, and Yahoo (indicated by the numeric codes 1,2,5,7). These four search engines are clustered together very tightly in all 6 query categories. The second cluster is composed of Google and AOL (numeric codes 4, 8) that also appears in very close proximity across all 6 query categories. The remaining three clusters contain individual engines: Live, Dogpile, and Ask.

The clusters in the embedding do in fact mirror the technology relationships that have evolved in the search engine industry. FAST, the company behind alltheweb,

bought Lycos and was subsequently bought by Overture who also bought Altavista[3]. Overture was subsequently bought by the fourth member of the cluster, Yahoo. All four search engines in the first cluster have close proximity in the embedding and yet are dissimilar from the remaining competitors. The second cluster, for Google and AOL, reflects the fact that AOL now relies heavily on Google's web search technology, leading to extremely similar ranked lists.

The remaining engines are quite distinct. Dogpile is a meta-search engine which incorporates the input of the other major search engines. We see that Dogpile's results are roughly equidistant from both Yahoo and Google clusters for all query categories. Figure 6 also shows that dogpile is more similar to the two remaining engines - Live and Ask. Apparently, dogpile emphasizes pages highly-ranked by Live and Ask in its meta search more than Google and AOL and more than Yahoo, Lycos, Altavista, and Alltheweb.

We present the similarity structure between the search engines in Figure 8. The figure displays a dendrogram output by standard hierarchical bottom up clustering. The clustering was computed based on the expected Hoeffding similarity measure for queries sampled from the query category companies. The dendrogram in the figure confirms the analysis above visually. Altavista, Alltheweb, Lycos, and Yahoo all form a tightly knit cluster. A similar tight cluster contains Google and AOL. More loosely it can also be said that Google and AOL are closer to the remaining search engines (Ask, Dogpile, and Live) than the Yahoo cluster.

3.4.2 Query Manipulations

In the second experiment we use search engine data to examine the sensitivity of the search engines to four commonly used query manipulation techniques. Assuming that the queries contained several words $w_1 w_2 \cdots w_l$ with $l > 1$, the query manipulation

[3] `http://google.blogspace.com/archives/000845.html`

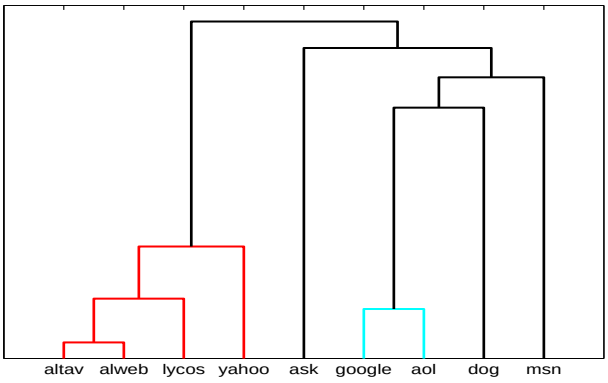


Figure 8: Hierarchical clustering dendrogram for the nine search engines over the query category Companies.

techniques that we consider are

$$
\begin{aligned}
&(a) \quad w_1 w_2 \cdots w_l \quad \Rightarrow \quad w_1 w_2 \cdots w_l \\
&(b) \quad w_1 w_2 \cdots w_l \quad \Rightarrow \quad w_1 + w_2 + \cdots + w_l \\
&(c) \quad w_1 w_2 \cdots w_l \quad \Rightarrow \quad \text{“}w_1 w_2 \cdots w_l\text{”} \\
&(d) \quad w_1 w_2 \cdots w_l \quad \Rightarrow \quad w_1 \text{ and } w_2 \text{ and } \cdots \text{ and } w_l \\
&(e) \quad w_1 w_2 \cdots w_l \quad \Rightarrow \quad w_1 \text{ or } w_2 \text{ or } \cdots \text{ or } w_l
\end{aligned}
$$

with the first technique (a) being the identity i.e. no query manipulation. The embeddings of queries in the query category celebrity are displayed in Figure 7. The left panel displays the MDS embedding based on our expected weighted Hoeffding distance with quadratic decaying weight. As a comparison, the right panel shows the MDS based on Kendall's top k distance as described by Fagin et al. [26]. Each marker in the figure represents the MDS embedding of a particular engine using a particular query manipulation technique which brings the total number of markers to $9 \cdot 5 = 45$.

Comparing the left and right panels, we see that visualizing using Kendall's top k distance [26] lacks discriminative power. The points in the right panel fall almost on top of each other limiting their use for visualization purposes. In contrast, the points in the left panel (weighted Hoeffding distance) differentiate not only different engines but also different types of query manipulations.

Table 5: The expected distance of different query manipulations from the original query for different search engines.

Engine/Distance	(b) +	(c) “”	(d) and	(e) or
Ask	0.5308	**0.6903**	0.6424	0.6447
Live	0.5625	**0.5639**	0.5006	0.5374
Google	0.3553	0.4117	**0.5584**	0.5500
Yahoo	0.4281	0.4647	0.5777	**0.5918**

In particular, it shows that most search engines produce two different clusters of results corresponding to two sets of query manipulation techniques: transformations {(a),(b),(c)} in one cluster and transformations {(d),(e)} in the other cluster. Live and ask form an exception to that rule forming clusters {(a),(d)}, {(b),(c)}, {(e)} (live) and {(a),(b),(d),(e)}, {(c)} (ask). Table 5 shows the query manipulations that produce ranked lists most distinct from the original query: (c) for Ask and Live, (d) for Google, and (e) for Yahoo.

3.4.3 Temporal Variation

In the third experiment, we visualize search result sets created by replicating 7 out of the 9 search engines over 7 consecutive days resulting in $7 \cdot 7 = 49$ search result sets. The search engines were queried on a daily basis during 3/25/2009 - 3/31/2009 and the returned results were embedded in 2-D for visualization. In contrast to the previous two experiments, we used a separate query category which was specifically aimed at capturing time sensitive matters. For example, we ignored tourism queries such as Eiffel Tower due to their time insensitive nature and instead used queries such as Timothy Geitner or AIG bonuses which dominated the news in March 2009. See Table 4 for more examples.

The embedded rankings are displayed in Figure 9 (top). The embedding reveals that the Yahoo cluster (Yahoo, Altavista, Alltheweb, and Lycos) shows a high degree of temporal variability, and in particular a sharp spatial shift on the third day from

the bottom region to the top left region. This could be interpreted either as a change in the index, reflecting the dynamic nature of the Web, or an internal change in the retrieval algorithms underlying the engines. The other engines are more stable as their ranked lists changed very little with the temporal variation. Note that as the queries are time sensitive this should not be interpreted as a measure of robustness, but rather as a stability measure for the internal index and ranking mechanisms. Interestingly, Vaughan [97] has also reported that Altavista shows temporal jumps with Google being more stable over a set of queries in 2004.

Figure 9 (bottom) shows the expected distance between the Yahoo search results across the seven consecutive days and a reference point (2nd and 6th day). As expected, for each one of the two plots, the deviation to the reference date increases monotonically with the temporal difference. The slope of the curve represents the degree of temporal change $\Delta(s_t, s_{t+\tau})$ between Yahoo at time t and at time $t+\tau$ as a function of τ with respect to the reference point t.

3.5 Application: Diversity of Search Results and Query Intents

For a given query, search engines aim to return ranked lists of documents or websites to cover diversified query intents at top-ranked positions. For example, in responding to the query 'office', search engines may return mixed websites about the TV show, the Microsoft product, or office supply shopping sites. The coverage and the ordering of query intents could be different depending on the internal mechanisms of the search engine. We are interested in answering the following questions: (a) how can we visualize ranking diversity for single or multiple search engines, given a set of queries representing the most common reformulations/intents? (b) how good is each engine at covering intents of an ambiguous query? (c) are differences in ranking between two search engines caused by differences in the intents they cover, or else if the aspects are similar, in the specific documents they choose to represent the intents? To help

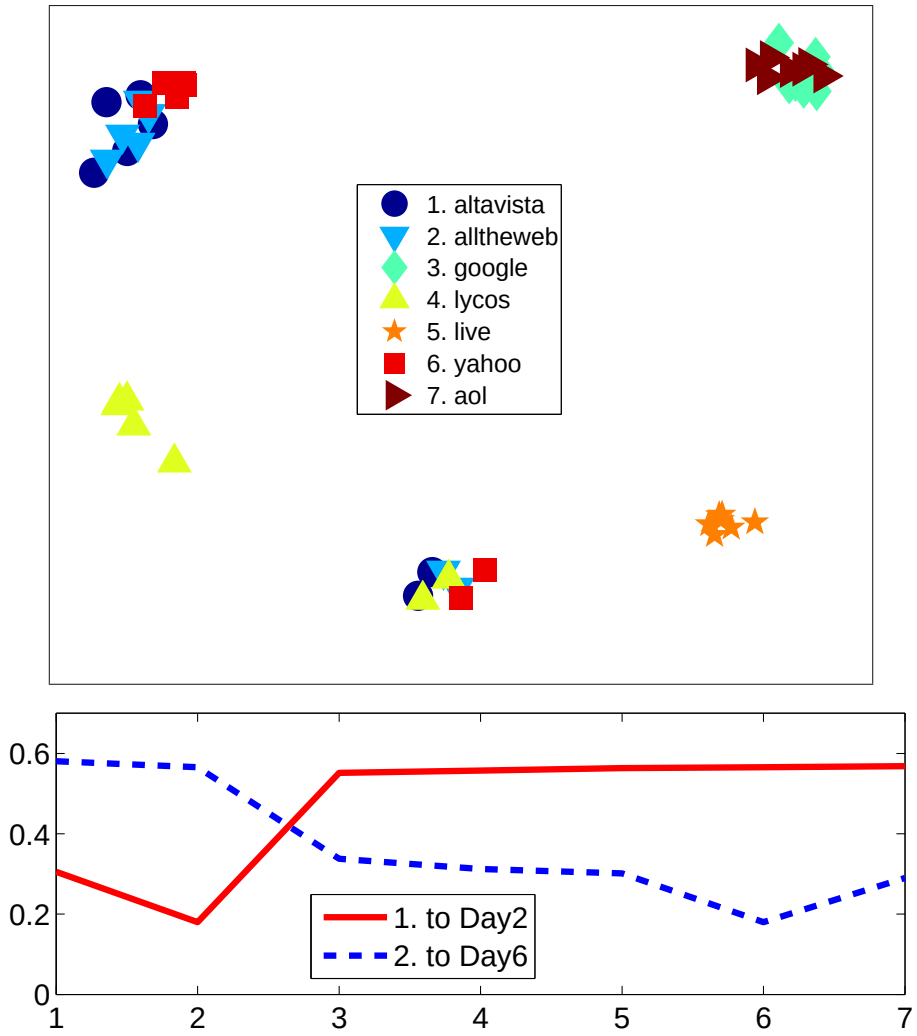


Figure 9: Top: MDS embedding of search engine results over seven days for a set of queries based on temporal events. The MDS embedding was based on the expected Hoeffding distance with linear weighting $w_t = 1/t$ over the top 50 sites. Bottom: The dissimilarity of Yahoo results over seven days with respective to a reference day for a set of queries based on temporal events.

answer these questions, we use visualizations based on Multi-Dimensional Scaling (MDS).

Since most search engines are optimized for the 'average user', we also consider the effect of varying a simple definition of an 'average user' and its expected rank distance to the (diverse) search engine results. We assess relative search engine diversity with respect to an 'average' user model using expected ranked-list distance. We also show how diversity effectiveness varies for each search engine with changes in the 'average' user model.

An *aggregated ranking* is a l-relevant partial ranked list $[j_1, \ldots, j_l]$, where the top l items are a tie and the other unranked $n-l$ items are a tie at the bottom. Following the computation in [91], the dissimilarity between a top-k ranking $\langle i_1, \ldots, i_k \rangle$ and the aggregated relevant ranking $[j_1, \ldots, j_l]$ can be computed in a closed form: For a top k and a relevant l list, the following closed form applies to the expected distance over the weighted Hoeffding distance (7).

$$\rho(\langle i_1, \ldots, i_k \rangle, [j_1, \ldots, j_l]) = \sum_{r=1}^{n} \bar{d}(r) \quad \text{where} \tag{21}$$

$$\bar{d}(r) = \begin{cases} \frac{1}{l} \sum_{t=1}^{l} d'_w(u,t) & r \in A \cap B \\ \frac{1}{n-l} \sum_{t=l+1}^{n} d'_w(u,t) & r \in A \cap B^c \\ \frac{1}{n-k} \frac{1}{l} \sum_{s=k+1}^{n} \sum_{t=1}^{l} d'_w(s,t) & r \in A^c \cap B \\ \frac{1}{n-k} \frac{1}{n-l} \sum_{s=k+1}^{n} \sum_{t=l+1}^{n} d'_w(s,t) & \text{otherwise} \end{cases}.$$

Above, $A = \{i_1, \ldots, i_k\}$, $B = \{j_1, \ldots, j_l\}$, and $u \in \{1, \ldots, k\}$ are the ranks of r in $\{i_1, \ldots, i_k\}$ (if they exist).

We illustrate the effectiveness of rank-distance measures and visualization for a set of 20 Web search queries, which we call 'initial' queries because they were chosen to be somewhat ambiguous in intent and often lead to a second, more specific reformulation that refines the intent. For each initial query, we obtained the most highly-weighted

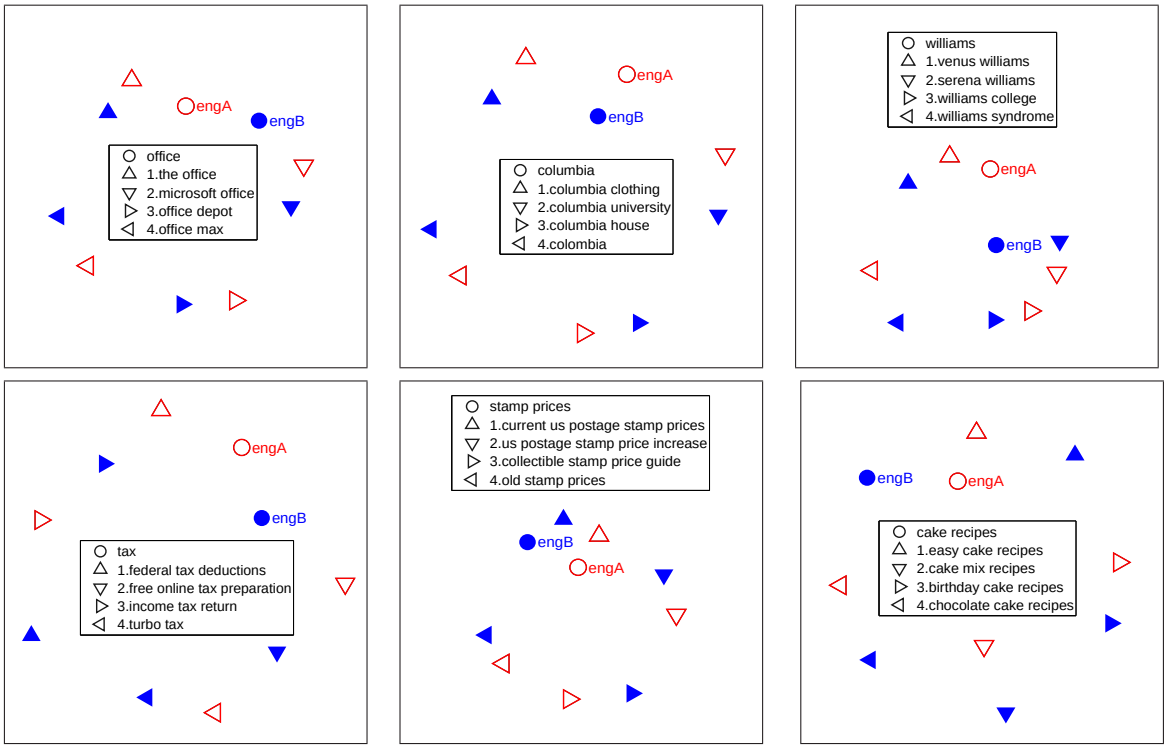


Figure 10: Diversity visualization using MDS of search result rankings for an initial query (circle) and its top intents (triangles), for 6 initial queries: 'office', 'columbia', 'williams', 'tax', 'stamp prices' and 'cake recipes' on two search engines. The MDS was based on the expected weighted Hoeffding distance with linear weighting $w_t = t^{-2}$ over the top 20 sites. Embeddings are scaled with respect to the top left query to preserve distance ratios.

subsequent query reformulations associated with the top intents using an algorithm that combines evidence from clicks and reformulations [75], based on the logs of a commercial search engine. For example, for an initial query 'columbia', reformulations for two popular query intents are 'columbia sports' and 'columbia university'. We will refer to these query reformulations and their corresponding results as 'intents' for brevity. Our query set is shown in Table 6, with an informal division into 'more' and 'less' ambiguous categories. For each initial query and the reformulated queries associated with its top intents, we retrieved the top 20 results from three major commercial search engines, which we label as 'Engine A', 'Engine B' and 'Engine C'.

3.5.1 Analyzing Differences in Diverse Rankings between Search Engines

In this section we ask the question: are differences in ranking between two search engines caused by differences in the aspects/intents they cover, or else if the aspects are similar, in the specific documents they choose to represent the intents?

The six visualizations in Figure 10 help answer this question for six example queries. Each boxed subfigure shows a two-dimensional MDS embedding of the distances between the ranked results for an initial query (circle) and its intent reformulations (triangles). Two different engine results are shown for comparison: Engine A (uncolored), and Engine B (solid). With this visualization, we can examine the diversity of the initial results with respect to the major intents for a single engine, or compare these across engines.

The diversity of the initial query result is reflected by the relative proximities to each intent, and reveals how a search engine may emphasize a particular intent over others. For example, for the initial query 'cake recipes', Engine A's results are much closer to the intent 'easy cake recipes' than Engine B's results, which have a more uniform distance to all intents (with slight preference toward 'chocolate cake recipes'). Similarly, we can see that the difference in results between A and B for the query 'office' (with an expected rank distance of 0.63) is mainly due to the order of the major intents between the two search engines. In particular, Engine A favors the intent 'the office', which is a TV show, while Engine B favors the intent 'microsoft office', a software product. In response to the ambiguous query 'williams', Engine A diversifies in favor of the tennis star 'venus williams' while Engine B's results prefer her sister 'serena williams'. We can also see that in general the two search engines tend to produce quite different results rankings for any initial or reformulated query.

For other queries, differences in ranking are due to differences in the order of specific documents and not on the relative allocation to different intents. For the query 'stamp prices', there is an expected distance of 0.48 between engines, but the

Table 6: The 20 queries used in our study, divided into More and Less Ambiguous classes.

More	ai, williams, live, office, columbia, house, cardinals, turkey, mercury, oscar, us open
Less	cake recipes, auto repair, search engines, currency tax,laptops, london travel, ancient rome, stamp prices

diversification among intents is very similar for both engines, with one major intent emphasized by both engines for 'current us postage stamp prices'. The visualization also shows that each intent's results are relatively similar between engines, and that some intents are similar, e.g. 'collectible stamp price guide' and 'old stamp prices'.

3.5.2 Measuring Ranking Diversity for Search Engines Compared to Aggregated Intents

In this experiment, we show how expected ranked-list distance can be used to perform a relative comparison of results diversity between engines. For each initial query, we construct an aggregated ranking (as defined in Section 3.5) from the results of all major query intents that represents the preferences of an 'average user'. In our simplified example, all intents are given equal weight, but this could be easily changed to support weighted intents. We then compute the expected weighted Hoeffding distance from the aggregate 'average user' ranking to each engine's results. The better a search engine's diversification matches the preferences of this average user, the smaller the distance will be. Results averaged over all 20 queries are shown in Table 7.

First, using expected distance to the aggregate 'average user' ranking effectively captures query ambiguity: distances for 'more ambiguous' queries are consistently smaller than those for the 'less ambiguous' queries – which is natural since 'more ambiguous' queries have results that cover a wider variety of intents, while less ambiguous queries tend to have a single predominating intent. Second, there is variation

Table 7: The distance between the ranking of an initial query and the aggregated ranking of major query intents for each engine, averaged over each group of more or less ambiguous queries. The top URLs for each query intent are weighted as equally relevant.

	EngineA	EngineB	EngineC
more ambiguous	0.7008	0.6882	0.7286
less ambiguous	0.7527	0.7658	0.7794

among the three search engines, with a significant difference between Engines A and B (overall averages 0.7267 and 0.7270) and Engine C (0.7540). One implication is that Engine C is diversifying in a way that is consistently different from A and B, using a less uniform subset of intents.

3.5.3 Assessing User-specific Diversity

One strength of the expected weighted Hoeffding distance is its ability to handle comparisons involving partial incomplete rankings in a principled way. This allows us to consider encoding a *user profile* that can contain preference information in the form of partial rankings of pages or domains we might have observed or inferred as part of a user's browsing or search history. The *personalized diversity score* (PDS) for a given (user, query) pair is the distance between the diversified search results ranking, and the ranking over whatever subset of the same pages or domains exist in the user's profile. When used with two or more search engines, the PDS provides a way to decide which engine's diversified results are most in accord with the user's preferences for a given query.

For a given query, we can also compute the average PDS of the ranked results over a set of potential user profiles, to assess how robust the diversity provided by the search engine is to uncertainty in what comprises the 'typical user'. Fig. 11 shows a smoothed density estimate of distances between search engine results and simulated users of different intents for a given query. Each simulated user is based on a complete

ordering of 4 URLs taken from the top results of 4 main intents, giving 24 possible profiles. The diversity in results for one engine is considered more robust compared to other engines if the former's distribution of distances to the user profile is shifted to the left (i.e. is consistently smaller across all profiles). We can see that there are queries for which there are indeed significant differences between engines. Using a two-sided Mann-Whitney U-test, all pairwise engine differences were significant at the $p = 0.05$ level for the queries 'columbia' and 'williams', and between Engine B and Engine C for the query 'office', with no difference for the query 'tax'.

To conclude, we show relative differences between the ranked results for initial ambiguous queries, and results for their major intents, for different commercial search engines. We also computed the expected distance between a search engine's results for an initial query, and an aggregated ranking of its major intents, to assess how queries and systems differed in the uniformity of their diversity coverage. We explored the effect of varying a simple user profile representing the 'average user' URL or domain preferences for a given query. In this way, we demonstrated queries for which there was a statistically significant difference between engines in their ability to provide a diversified ranking that was robust to different hypotheses of the 'average user'.

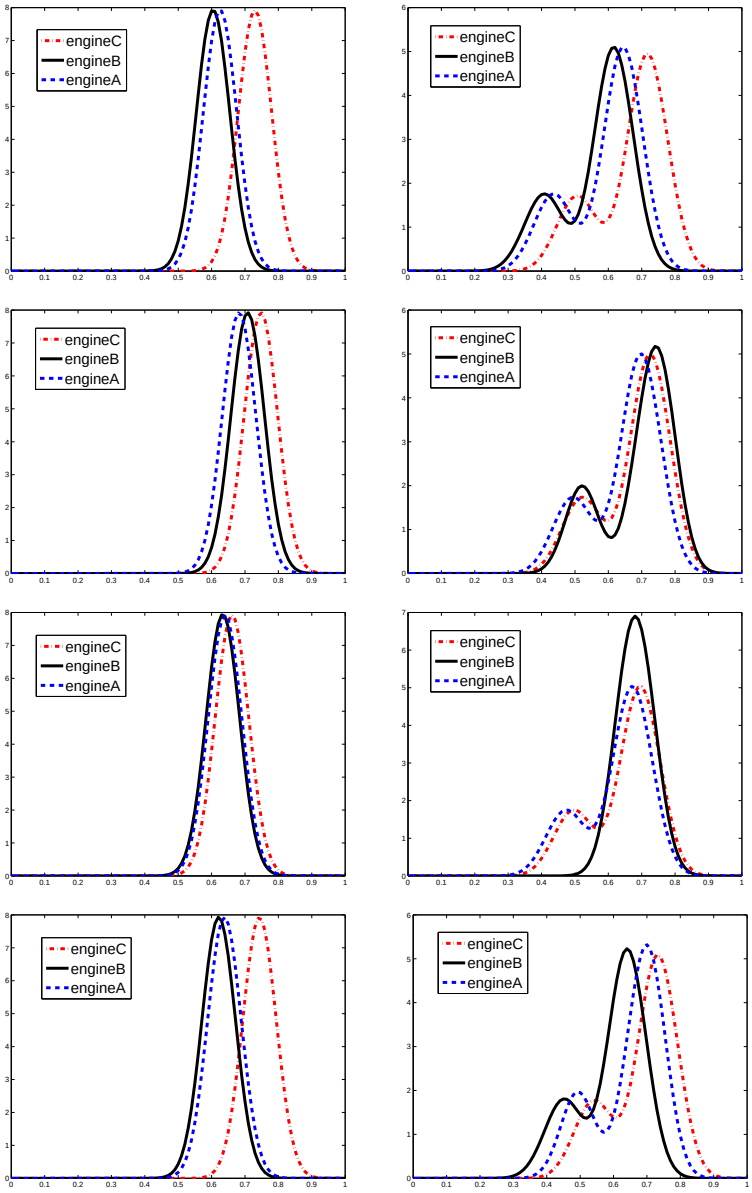


Figure 11: Distributions of user-relative diversity scores for four queries, computed by uniformly sampling from the same set of potential user profiles for all engines, and x-axis gives expected rank distance between the search engine's top-ranked results and the preference ranking expressed in the user profile. From top to bottom, the queries are 'columbia', 'office', 'williams' and 'tax'. The distance adopts the weight function $w = 1$ in the left column and $w = 1/k$ topK weighted in right column.

CHAPTER IV

MODELING PARTIAL INCOMPLETE RANKINGS

Modeling incomplete rankings is very important from a practical perspective and yet has not been fully studied due to considerable computational difficulties. To avoid such computational difficulties and efficiently, we propose to construct a non-parametric model with triangular kernel smoothing for rankings with missing items. We demonstrate our approach and show how it applies in the context of collaborative filtering.

4.1 Related Work

There is a significant body of work on modeling ranking data in statistics. Most of those work has focused on simple generative models estimating a parametric distribution of permutations or partial rankings. Early work in this direction includes the Thurstone model [94] and the Babington Smith model [87]. A special case of the Babington Smith model is the Mallows [63], which appears as analogue of the Gaussian distributions. However, the distribution of rankings does not follow a simple parametric form. Instead, the distribution tends to be a diffuse and multimodal distribution with different probability mass regions corresponding to different types of judges.

Since 1980, a broad range of modeling approaches from different perspectives are developed. Fligner and Verducci [27, 28] have proposed a generalization of the Mallows model for multistage rankings, followed with a discussion of posterior probabilities estimation in [29]. Diaconis has approached the problem using group representations with emphasis on analysis of variance methods [23, 24]. A unified view for partial rankings is presented by Critchlow [20].

In the last past decade, a new surge of interest in ranking data emerges in the area of machine learning and data mining. Early attempts bridging the statistics models and machine learning algorithms include [57, 58], in which generalizations of the Mallows model are proposed for estimating conditional distributions. A different class of estimators is based on Fourier analysis on the symmetric group [23, 53, 44], focusing on maintaining and updating a distribution over permutations by a low frequency approximation.

Most of these techniques are designed to work with fully ranked data. The few possible exceptions [20, 65] can be applied to partial rankings, although the models are ill suited to handle partial rankings of different types. Other attempts include learning mixtures of Mallows models [12] for top-k rankings and the hierarchical riffle independent models [45]. Many of those approaches have not addressed the scalability problem for large n. Alternatively, a non-parametric model assuming a Mallows kernel has been proposed by Lebanon and Mao in [59], where efficient computational procedures are developed for tied rankings.

Instead of using Mallows kernel, a triangular kernel is developed in [50] to apply the density estimation for incomplete rankings. However, the computational procedure is inefficient to estimate the probability of incomplete rankings of arbitrary structures. It is suggested in [50] to compute the probability of a full ranking from incomplete rankings using normal approximations for distance distributions. However, estimating the probability of incomplete rankings from incomplete rankings in general scenarios remains an open problem.

In this work, we propose an efficient non-parametric smoothing framework using triangular kernel for estimating probabilities of incomplete rankings. This work extends the non-parametric density estimation approach over rankings in [50] and makes it amenable to a wide range of real world applications. This technique may be used in recommendation systems in different ways. It provides a statistically

meaningful estimation framework for issuing recommendations in conjunction with decision theory. Moreover, it also leads to other important applications including mining association rules, exploratory data analysis, and clustering items and users. Two key observations that we make are: (i) incomplete tied preference data may be interpreted as randomly censored permutation data, and (ii) generating functions combined with non-zero smoothing or modified triangular kernel are able to provide a computationally efficient scheme for computing the estimator.

We proceed in the next sections to describe notations and our assumptions and estimation procedure, and follow with case studies demonstrating our approach on real world recommendation system data.

4.2 Definitions and Estimation Framework

Formally, we have m users providing incomplete tied preference relations on n items

$$\begin{aligned} S_1: &\quad A_{1,1} \prec A_{1,2} \prec \cdots \prec A_{1,k(1)} \\ S_2: &\quad A_{2,1} \prec A_{2,2} \prec \cdots \prec A_{2,k(2)} \\ &\quad \vdots \\ S_m: &\quad A_{m,1} \prec A_{m,2} \prec \cdots \prec A_{m,k(m)} \end{aligned} \tag{22}$$

where $A_{i,j} \subset \{1, \ldots, n\}$ are sets of items (wlog we identify items with integers $1, \ldots, n$) defined by the following interpretation: user i prefers all items in $A_{i,j}$ to all items in $A_{i,j+1}$. The notation $k(i)$ above is the number of such sets provided by user i. The data (22) are incomplete since not all items are necessarily observed by each user i.e., $\bigcup_{j=1}^{k(i)} A_{i,j} \subsetneq \{1, \ldots, n\}$ and may contain ties since some items are left uncompared, i.e., $|A_{i,j}| > 1$. Recommendation systems recommend items to a new user, denoted as $m+1$, based on their preference

$$S_{m+1} : A_{m+1,1} \prec A_{m+1,2} \prec \cdots \prec A_{m+1,k(m+1)} \tag{23}$$

and its relation to the preferences of the m users (22).

As an illustrative example, assuming $n = 9, m = 3$, the data

$$\begin{aligned} S_1 : &\qquad 1, 8, 9 \prec 4 \prec 2, 3, 7 \\ S_2 : &\qquad 4 \prec 2, 3 \prec 8 \\ S_3 : &\qquad 4, 8 \prec 2, 6, 9 \end{aligned}$$

correspond to $A_{1,1} = \{1, 8, 9\}$, $A_{1,2} = \{4\}$, $A_{1,3} = \{2, 3, 7\}$, $A_{2,1} = \{4\}$, $A_{2,2} = \{2, 3\}$, $A_{2,3} = \{8\}$, $A_{3,1} = \{4, 8\}$, $A_{3,2} = \{2, 6, 9\}$, and $k(1) = k(2) = 3, k(3) = 2$. From the data we may guess that item 4 is relatively popular across the board while some users like item 8 (users 1, 3) and some hate it (user 2). Given a new $m + 1$ user issuing the preference $1 \prec 2, 3, 7$ we might observe a similar pattern of preference or taste as user 1 and recommend to the user item 8. We may also recommend item 4 which has broad appeal resulting in the augmentation

$$1 \prec 2, 3, 7 \qquad \mapsto \qquad 1, 4, 8 \prec 2, 3, 7.$$

We note that in some cases the preference relations (22) arise from users providing numeric scores to items. For example, if the users assign 1-5 stars to movies, the set $A_{i,j}$ contains all movies that user i assigned $6 - j$ stars to and $k(i) = 5$ (assuming some movies were assigned to each of the 1, 2, 3, 4, 5 star levels). As pointed out by a wide variety of studies in economics and social sciences, e.g. [15], such numeric scores are inconsistent among different users. We therefore proceed to interpret such data as ordinal rather than numeric.

We describe the following notations and conventions for permutations, which are taken from [23] where more detail may be found. We denote a permutation by listing the items from most preferred to least separated by a $\prec$ or $|$ symbol: $\pi^{-1}(1) \prec \pi^{-1}(2) \prec \cdots \prec \pi^{-1}(n)$, e.g. $\pi(1) = 2, \pi(2) = 3, \pi(3) = 1$ is $3 \prec 1 \prec 2$. Ranking with ties occur when judges do not provide enough information to construct a total order. In particular, we define tied rankings as a partition of $\{1, \ldots, n\}$ to $k < n$ disjoint

subsets $A_1, \ldots, A_k \subset \{1, \ldots, n\}$ such that all items in A_i are preferred to all items in A_{i+1} but no information is provided concerning the relative preference of the items among the sets A_i. We denote such rankings by separating the items in A_i and A_{i+1} with a $\prec$ or $|$ notation. For example, the tied ranking $A_1 = \{3\}, A_2 = \{2\}, A_3 = \{1, 4\}$ (items 1 and 4 are tied for last place) is denoted as $3 \prec 2 \prec 1, 4$ or $3|2|1, 4$.

Ranking with missing items occur when judges omit certain items from their preference information altogether. For example assuming a set of items $\{1, \ldots, 4\}$, a judge may report a preference $3 \prec 2 \prec 4$, omitting altogether item 1 which the judge did not observe or experience. This case is very common in situations involving a large number of items n. In this case judges typically provide preference only for the $l \ll n$ items that they observed or experienced. For example, in movie recommendation systems we may have $n \sim 10^3$ and $l \sim 10^1$.

Rankings can be full (permutations), with ties, with missing items, or with both ties and missing items. In either case we denote the rankings using the $\prec$ or $|$ notation or using the disjoint sets $A_1, \ldots, A_k$ notation. We also represent tied and incomplete rankings by the set of permutations that are consistent with it. For example,

$$3 \prec 2 \prec 1, 4 = \{3 \prec 2 \prec 1 \prec 4\} \cup \{3 \prec 2 \prec 4 \prec 1\}$$

$$3 \prec 2 \prec 4 = \{1 \prec 3 \prec 2 \prec 4\} \cup \{3 \prec 1 \prec 2 \prec 4\} \cup \{3 \prec 2 \prec 1 \prec 4\} \cup \{3 \prec 2 \prec 4 \prec 1\}$$

are sets of two and four permutations corresponding to tied and incomplete rankings, respectively.

It is hard to directly posit a coherent probabilistic model on incomplete tied data such as (22). Different preferences relations are not unrelated to each other: they may subsume one another (for example $1 \prec 2 \prec 3$ and $1 \prec 3$), represent disjoint events (for example $1 \prec 3$ and $3 \prec 1$), or interact in more complex ways (for example $1 \prec 2 \prec 3$ and $1 \prec 4 \prec 3$). A valid probabilistic framework needs to respect the constraints resulting from the axioms of probability, e.g., $p(1 \prec 2 \prec 3) \leq p(1 \prec 3)$.

Our approach is to consider the incomplete tied preferences as censored permutations. That is, we assume a distribution $p(\pi)$ over permutations $\pi \in \mathfrak{S}_n$ ($\mathfrak{S}_n$ is the symmetric group of permutations of order n) that describes the complete withouties preferences in the population. The data available to the recommender system (22) are sampled by drawing m iid permutations from p: $\pi_1, \ldots, \pi_m \overset{\text{iid}}{\sim} p$, followed by censoring to result in the observed preferences $S_1, \ldots, S_m$

$$\pi_i \sim p(\pi), \quad S_i \sim p(S|\pi_i), \quad i = 1, \ldots, m+1 \tag{24}$$

$$p(\pi|S) = \frac{I(\pi \in S)p(\pi)}{\sum_{\sigma \in S} p(\sigma)} \tag{25}$$

$$p(S|\pi) = p(\pi|S)q(S)/p(\pi) = \frac{I(\pi \in S)p(\pi)q(S)}{p(\pi)\sum_{\sigma \in S} p(\sigma)} = \frac{I(\pi \in S)q(S)}{\sum_{\sigma \in S} p(\sigma)} \tag{26}$$

where $q(S)$ represents the probability of observing the censoring S consisting of permutations σ or equivalently it describes a random process resulting in a particular censoring (specifically, it is not equal to $\sum_{\sigma \in S} p(\sigma)$).

Although many approaches for estimating p given $S_1, \ldots, S_m$ are possible, experimental evidence point to the fact that in recommendation systems with high n, the distribution p does not follow a simple parametric form such as the Mallows, Bradley-Terry, or Thurstone models [65]. See Figure 2 for a demonstration how the number of modes and complexity increase with n. In this figure, which appears also in [51], we show a density estimate (using kernel smoothing) of rankings embedded in a two dimensional space using multidimensional scaling. The distance function in this case was the average Kendall's tau distance over all possible permutations consistent with the partial rankings. The figure shows three different panels corresponding to different data sets of increasing n. As n increases, the number of the modes increases and the density surface itself becomes less regular. Intuitively, different probability mass regions correspond to different types of judges. For example in movie preferences probability modes may correspond to genre as fans of drama, action, comedy, etc. having similar preferences.

We therefore propose to estimate the underlying distribution p on permutations extending non-parametric kernel smoothing on rankings [59]. The standard kernel smoothing formula applies to the permutation setting as

$$\hat{p}(\pi) = \frac{1}{m} \sum_{i=1}^{m} K_h(T(\pi, \pi_i))$$

where $\pi_1, \ldots, \pi_m \overset{\text{iid}}{\sim} p$, T a distance on permutations such as Kendall's distance and $K_h(r) = h^{-1}K(r/h)$ a normalized unimodal function. In the case at hand, however, the observed preferences π_i as well as π are replaced with permutations sets $S_1, \ldots, S_m, R$ representing incomplete tied preferences

$$\hat{p}(R) = \sum_{\pi \in R} \hat{p}(\pi) = \frac{1}{m} \sum_{i=1}^{m} \sum_{\pi \in R} \sum_{\sigma \in S_i} q(\sigma|S_i) K_h(T(\pi, \sigma)) \tag{27}$$

where $q(\sigma|S_i)$ serves as a surrogate for the unknown $p(\sigma|S_i) \propto I(\sigma \in S_i)p(\sigma)$ (see (25)). For example, a uniform $q(\sigma|S_i)$ indicates that given a censored ranking corresponding to a user's ratings, the precise permutation of preferences is uniform over the set of compatible permutations.

Selecting $q(\sigma|S_i) = p(\sigma|S_i)$ would lead to consistent estimation of $p(R)$ in the limit $h \to 0$, $m \to \infty$ assuming positive $p(\pi), p(S)$ by appealing to standard kernel density consistency results found in [100]. Such a selection, however, is generally impossible since $p(\pi)$ and therefore $p(\sigma|S_i)$ is unknown.

In general the specific choice of the surrogate $q(\sigma|S)$ is important as it may influence the estimated probabilities. Furthermore, it may cause underestimation or overestimation of $\hat{p}(R)$ in the limit of large data. An exception occurs when the sets $S_1, \ldots, S_m$ are either subsets of R or disjoint from R. In this case $\lim_{h \to 0} K_h(\pi, \sigma) = I(\pi = \sigma)$ resulting in the following limit (with probability 1 by the strong law of large

numbers)

$$\lim_{m\to\infty}\lim_{h\to 0}\hat{p}(R) = \lim_{m\to\infty}\frac{1}{m}\sum_{i=1}^{m} I(S_i \subset R)\sum_{\sigma\in S_i} q(\sigma|S_i) \tag{28}$$

$$= \lim_{m\to\infty}\frac{1}{m}\sum_{i=1}^{m} I(S_i \subset R) = \lim_{m\to\infty}\frac{1}{m}\sum_{i=1}^{m} I(\pi_i \in R) = p(R). \tag{29}$$

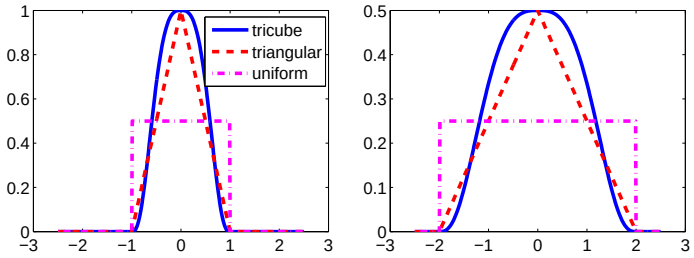
Thus, if our data are comprised of preferences S_i that are either disjoint or a subset of R we have consistency *regardless* of the choice of the surrogate q. Such a situation is more realistic when the preference R involves a small number of items and the preferences S_i, $i = 1, \ldots, m$ involve a larger number of items. This is often the case for recommendation systems where individuals report preferences over 10-100 items and we are mostly interested in estimating probabilities of preferences over fewer items such as $i \prec j, k$ or $i \prec j, k \prec l$ (see experiment section). Nevertheless, real world recommendation systems data may show sparsity patterns that violate this assumption. In such cases the proposed method may still be used for engineering purposes but the consistency result no longer applies.

The main difficulty with the estimator above is the computational complexity of the equation $\sum_{\pi\in R}\sum_{\sigma\in S_i} q(\sigma|S_i)K_h(T(\pi,\sigma))$. In the case of high n and only a few observed items k the sets S_i, R grow factorially as $(n-k)!$ making a naive computation of (27) intractable for all but the smallest n. In the next section we explore efficient computations of these sums for a triangular kernel K_h and a uniform $q(\pi|S)$.

4.3 Computationally Efficient Kernel Smoothing

In previous work [59] the estimator (27) is proposed for tied (but complete) rankings. That work derives closed form expressions and efficient computation for (27) assuming a Mallows kernel [63]

$$K_h(T(\pi,\sigma)) = \exp\left(-\frac{T(\pi,\sigma)}{h}\right)\prod_{j=1}^{n}\frac{1-e^{-1/h}}{1-e^{-j/h}} \tag{30}$$



	$K_3(\cdot, 1 \prec 2 \prec 3)$	$K_5(\cdot, 1 \prec 2 \prec 3)$
$1 \prec 2 \prec 3$	0.50	0.33
$1 \prec 3 \prec 2$	0.25	0.22
$2 \prec 1 \prec 3$	0.25	0.22
$3 \prec 1 \prec 2$	0	0.11
$2 \prec 3 \prec 1$	0	0.11
$3 \prec 2 \prec 1$	0	0

Figure 12: Tricube, triangular, and uniform kernels on $\mathbb{R}$ with bandwidth $h = 1$ (left) and $h = 2$ (middle). Right: triangular kernel on permutations ($n = 3$).

where T is Kendall's tau distance on permutations (below $I(x) = 1$ for $x > 0$ and 0 otherwise)

$$T(\pi, \sigma) = \sum_{i=1}^{n-1} \sum_{l>i} I(\pi\sigma^{-1}(i) - \pi\sigma^{-1}(l)). \tag{31}$$

Unfortunately these simplifications do not carry over to the case of incomplete rankings where the sets of consistent permutations $S_1, \ldots, S_m$ are not cosets of the symmetric group. As a result the problem of probability estimation in recommendation systems where n is high and many items are missing is particularly challenging. Alternatively, replacing the Mallows kernel (30) with a triangular kernel leads to efficient computation in the case of ties and incomplete rankings. Specifically, the triangular kernel on permutation, which is also adopted in [50], is

$$K_h(T(\pi, \sigma)) = (1 - h^{-1}T(\pi, \sigma))\, I(h - T(\pi, \sigma))\, / \, C \tag{32}$$

where the bandwidth parameter h represent both the support (the kernel is 0 for all larger distances) and the inverse slope of the triangle. The normalization term C is a function of h and may be efficiently computed using generating functions [50]. Figure 12 (right panel) displays the linear decay of (32) for the simple case of permutations over $n = 3$ items.

Generating functions, a tool from enumerative combinatorics, allow efficient computation of (27) by concisely expressing the distribution of distances between permutations. Kendall's tau $T(\pi, \sigma)$ is the total number of discordant pairs or inversions

between π, σ and thus its computation becomes a combinatorial counting problem. The generating function with the symmetric group of permutations of n items [90] is the following:

$$G_n(z) = \prod_{j=1}^{n-1} \sum_{k=0}^{j} z^k. \tag{33}$$

As shown in [90], the coefficient of z^k of $G_n(z)$, which we denote as $[z^k]G_n(z)$, corresponds to the number of permutations σ for which $T(\sigma, \pi') = k$. For example, the distribution of Kendall's tau $T(\cdot, \pi')$ over all permutations of 3 items is described by $G_3(z) = (1+z)(1+z+z^2) = 1z^0 + 2z^1 + 2z^2 + 1z^3$ i.e., there is one permutation σ with $T(\sigma, \pi') = 0$, two permutations σ with $T(\sigma, \pi') = 1$, two with $T(\sigma, \pi') = 2$ and one with $T(\sigma, \pi') = 3$. Another important generating function is

$$H_n(z) = \frac{G_n(z)}{1-z} = (1 + z + z^2 + z^3 + \cdots)G_n(z)$$

where $[z^k]H_n(z)$ represents the number of permutations σ for which $T(\sigma, \pi') \leq k$.

Proposition 5. *The normalization term $C(h)$ is given by $C(h) = [z^h]H_n(z) - h^{-1}[z^{h-1}]\frac{G_n'(z)}{1-z}$.*

Proof. The proof factors the non-normalized triangular kernel $CK_h(\pi, \sigma)$ to $I(h - T(\pi, \sigma))$ and $h^{-1}T(\pi, \sigma)I(h - T(\pi, \sigma))$ and making the following observations. First we note that summing the first factor over all permutations may be counted by $[z^h]H_n(z)$. The second observation is that $[z^{k-1}]G_n'(z)$ is the number of permutations σ for which $T(\sigma, \pi') = k$, multiplied by k. Since we want to sum over that quantity for all permutations whose distance is less than h we extract the $h-1$ coefficient of the generating function $G_n'(z)\sum_{k\geq 0} z^k = G_n'(z)/(1-z)$. We thus have

$$C = \sum_{\sigma: T(\pi', \sigma) \leq h} 1 - h^{-1} \sum_{\sigma: T(\pi', \sigma) \leq h} T(\pi', \sigma) = [z^h]H_n(z) - h^{-1}[z^{h-1}]\frac{G_n'(z)}{1-z}.$$

□

Proposition 6. *The complexity of computing $C(h)$ is $O(n^4)$.*

Proof. We describe a dynamic programming algorithm to compute the coefficients of G_n by recursively computing the coefficients of G_k from the coefficients of G_{k-1}, $k = 1, \ldots, n$. The generating function $G_k(z)$ has $k(k+1)/2$ non-zero coefficients and computing each of them (using the coefficients of G_{k-1}) takes $O(k)$. We thus have $O(k^3)$ to compute G_k from G_{k-1} which implies $O(n^4)$ to compute G_k, $n = 1, \ldots, n$. We conclude the proof by noting that once the coefficients of G_n are computed the coefficients of $H_n(z)$ and $G_n(z)/(1-z)$ are computable in $O(n^2)$ as these are simply cumulative weighted sums of the coefficients of G_n. □

Note that computing $C(h)$ for one or many h values may be done offline prior to the arrival of the rankings and the need to compute the estimated probabilities.

Denoting by k the number of items ranked in either S or R or both, the computation of $\hat{p}(\pi)$ in (27) requires $O(k^2)$ online and $O(n^4)$ offline complexity if either non-zero smoothing is performed over the entire data i.e., $\max_{\pi \in R} \max_{i=1}^{n} \max_{\sigma \in S_i} T(\sigma, \pi) < h$ or alternatively, we use the modified triangular kernel $K_h^*(\pi, \sigma) \propto (1 - h^{-1})T(\pi, \sigma)$ which is allowed to take negative values for the most distant permutations (normalization still applies though). A pseudo-code description of the probability computation is given as Algorithm 2.

Corollary 1. *Denoting the number of items ranked by either S or R or both as k, and assuming either $h > \max_{\pi \in R} \max_{i=1}^{n} \max_{\sigma \in S_i} T(\sigma, \pi)$ or that the modified triangular kernel $K_h^*(\pi, \sigma) \propto (1 - h^{-1})T(\pi, \sigma)$ is used, the complexity of computing $\hat{p}(R)$ in (27) (assuming uniform $q(\pi|S_i)$) is $O(mk^2)$ online and $O(n^4)$ offline.*

Proof. The proof follows from noting that (27) reduces to $O(n^4)$ offline computation of the normalization term, $O(k^2)$ online computation of the form (6) and $O(m)$ computation of the final summation. □

Algorithm 2 Algorithm to compute the probability of partial incomplete ranking R given the population $S_1, \ldots, S_m$ as described in equation (27). The online complexity of the algorithm is $O(mk^2)$ as described in Corollary 1.

Off-line:

1. Specify n, the number of total items and h the kernel bandwidth.
2. Precompute the normalization term $C(h)$ as in Proposition 5.

On-line:

3. Call Probability-Partial-Incomplete(R, S_1,...,S_m) for ranking R given $S_1, \ldots, S_m$.

function Probability-Partial-Incomplete(R, $S_1, \ldots, S_m$)
$sum = 0$;
for $i = 1 \to m$ **do**
  $\bar{d}$ = Expected-Kendall(R, S_i)
  Update $sum = sum + |R| K_h(\bar{d})$ using equation (32).
end for
$sum = sum/m$
return sum

function Expected-Kendall(R,S)
Label $1, \ldots, k$ as the unique items ranked in R or S.
Compute the minimum rank and number of ties τ_R, ϕ_R, τ_S, and ϕ_S for R and S.
$sum = 0$
for $i = 1 \to k$ **do**
  for $j = i + 1 \to k$ **do**
    $p_{ij}(R)$ = Probability-Pair(i, j, τ_R, ϕ_R); $p_{ij}(S)$ = Probability-Pair(i, j, τ_S, ϕ_S)
    $sum = sum + (1 - 2p_{ij}(R))(1 - 2p_{ij}(S))$
  end for
  $j = k + 1$
  $p_{ij}(R)$ = Probability-Pair(i, j, τ_R, ϕ_R); $p_{ij}(S)$ = Probability-Pair(i, j, τ_S, ϕ_S)
  $sum = sum + (n - k)(1 - 2p_{ij}(R))(1 - 2p_{ij}(S))$
end for
$sum = n(n-1)/4 - sum/2$
return sum

function Probability-Pair(i,j,τ_U,ϕ_U)
$p = 0$
if i and j are ranked in U with $\tau_U(i) = \tau_U(j)$ **then**
  $p = 1/2$
else if i and j are ranked in U with $\tau_U(i) \neq \tau_U(j)$ **then**
  $p = I(\tau_U(j) - \tau_U(i))$
else if only i is ranked in U **then**
  $p = 1 - \frac{\tau_U(i) + \frac{\phi_U(i)-1}{2}}{k+1}$
else if only j is ranked in U **then**
  $p = \frac{\tau_U(j) + \frac{\phi_U(j)-1}{2}}{k+1}$
else
  $p = 1/2$
end if
return p

4.4 Applications and Case Studies

We will demonstrate how to apply our estimator to collaborative filtering recommendation systems. A substantial body of literature in computer science has addressed the problem of constructing recommendation systems as this has been an active research area since the 1990s. We have attempted to outline the most important and successful approaches. The earliest efforts made a prediction for the rating of items based on the similarity of the test user and the training users [79, 10, 40]. Specifically, these attempts used similarity measures such as Pearson correlation [79] and Vector cosine similarity [10, 40] to evaluate the similarity level between different users. Other memory-based approaches include item-item similarities [84] and Slope-one [61].

Model-based approaches include user and item clustering [10, 14, 18, 96, 101], Bayesian classifiers [68, 69], regression-based methods [99], Bayesian networks [10], dependence network [39] and probabilistic latent variable models [72, 43, 66, 71, 103].

Most recently, the state of the art methods including the winner of the Netflix competition are based on non-negative matrix factorization of the partially observed user-rating matrix. The factorized matrix can be used to fill out the unobserved entries in a way similar to latent factor analysis [34, 78, 56, 54, 60, 83, 82, 55, 89, 78, 22].

Each of the above methods focuses exclusively on user ratings. In some cases item information is available (movie genre, actors, directors, etc) which have lead to several approaches that combine voting information with item information [7, 73, 85].

Our method differs from the methods above in that it constructs a full probabilistic model on preferences. It is able to handle heterogeneous preference information (not all users must specify the same number of preference classes) and does not make any parametric assumptions. In contrast to previous approaches it enables clear meaningful statistical estimation procedures for not only the prediction of item ratings,

but also the discovery of association rules and the estimation of probabilities of interesting events. Note that non-negative matrix factorization may be considered a probabilistic model assuming exponential family models such as Poisson or Normal. Such an approach, however, models the scores as numeric variables rather than the ordering themselves as is the approach of this work.

We divide our experimental study to three parts. In the first we examine the task of predicting probabilities. The remaining two parts use these probabilities for rank prediction and rule discovery. Motivation for the multiple evaluation paradigms used can be found in both the probabilistic nature of the estimated rankings and the widely acknowledge difficulty in the evaluation of recommender systems [41].

In our experiments we use three data sets. The MovieLens data set[1] contains one million ratings from 6040 users on 3952 movies with a scale of 1-5. The EachMovie data set[2] contains 2.6 million ratings from 74424 users on 1648 movies with a scale of 0-5. The Netflix data set [3] contains 100 million movie ratings from 480189 users on 17770 movies with a scale of 1-5. In all of these data sets users typically rated only a small number of items. Histograms of the distribution of the number of votes per user, number of votes per item, and vote distribution appear in Figure 13. Details of sampling and partitioning the data sets to suit different tasks are described in each section.

4.4.1 Estimating Probabilities

We consider here the task of estimating $\hat{p}(R)$ where R is a set of permutations corresponding to a tied incomplete ranking. Such estimates may be used to compute conditional estimates $\hat{P}(R|S_{m+1})$ which are used to predict which augmentations R of S_{m+1} are highly probable. For example, given an observed preference $3 \prec 2 \prec 5$ we

[1] http://www.grouplens.org
[2] http://www.grouplens.org/node/76
[3] http://www.netflixprize.com/community

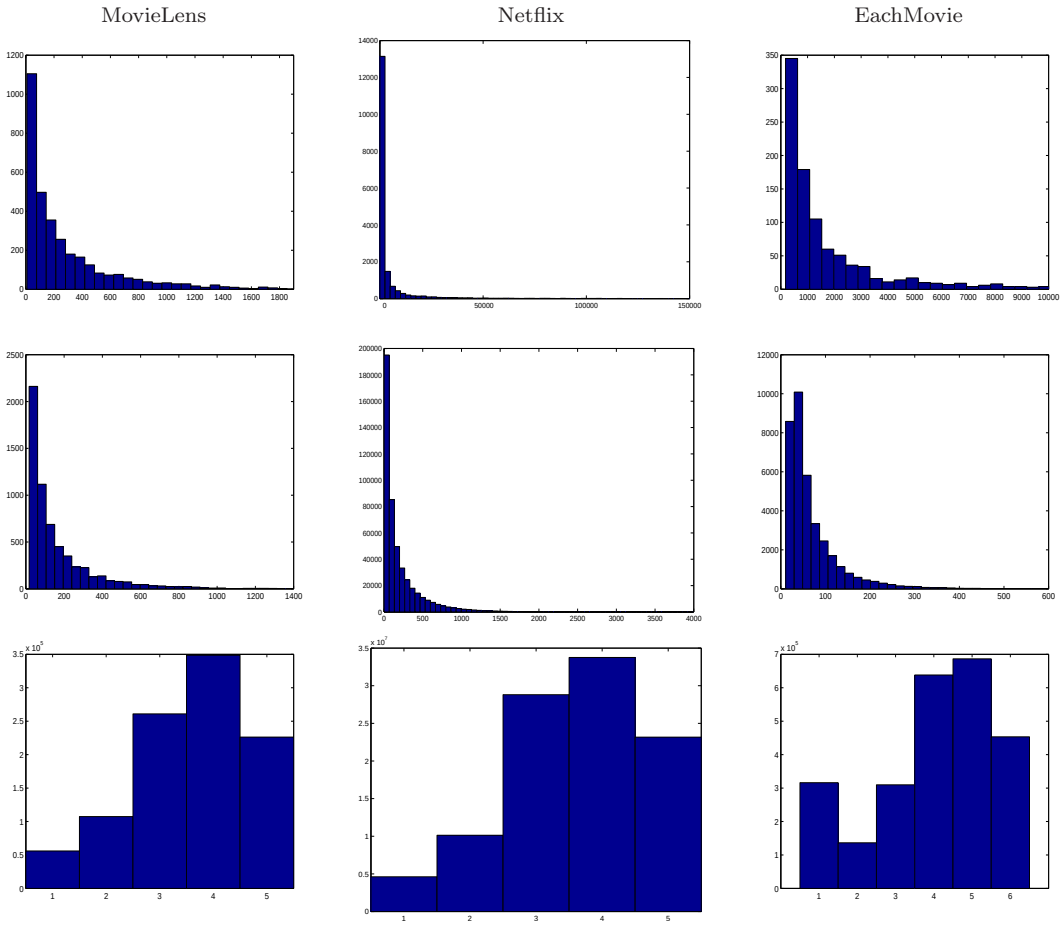


Figure 13: Histograms of the number of user votes per movie (top row), number of movies ranked per user (middle row), and votes aggregated over all users and movies (bottom row).

may want to compute $\hat{p}(8 \prec 3 \prec 2 \prec 5 | 3 \prec 2 \prec 5) = \hat{p}(8 \prec 3 \prec 2 \prec 5)/\hat{p}(3 \prec 2 \prec 5)$ to see whether item 8 should be recommended to the user.

For simplicity we focus in this section on probabilities of simple events such as $i \prec j$ or $i \prec j \prec k$. The next section deals with more complex events. In our experiment, we estimate the probability of $i \prec j$ for the $n = 53$ most rated movies in Netflix from $m = 10000$ users who rate most of these movies. We normalize the Kendall's tau distance for rankings of n items by dividing the maximum distance $n(n-1)/2$ and set the kernel bandwidth parameter $h = 0.38$. The probability matrix of the pairs is shown in Figure 14 where each cell corresponds to the probability of preference between a pair of movies determined by row j and column i. In the top left panel the rows and columns are ordered by average probability of a movie being preferred to others $r(i) = \frac{\sum_j \hat{p}(i \prec j)}{n}$ with the most preferred movie in row and column 1 (top right panel indicates the ordering according to $r(i)$). It is interesting to note the high level of correlation between the average probabilities and the pairwise probabilities as indicated by the uniform color gradient. In the bottom left panel the movies were ordered first by popularity of genres and then by $r(i)$. The bottom right panel indicates that ordering. The names, genres, and both orderings of all 53 movies appear in Table 8.

The three highest movies in terms of $r(i)$ are Lord of the Rings: The Return of the King, Finding Nemo, and Lord of the Rings: The Two Towers. The three lowest movies are Maid in Manhattan, Anger Management, and The Royal Tenenbaums. Examining the genre (colors in right panels of Figure 14) we see that family and science fiction are generally preferred to others movies while comedy and romance generally receive lower preferences. The drama, action genres are somewhere in the middle.

Also interesting is the variance of the movie preferences within specific genres. Family movies are generally preferred to almost all other movies. Science fiction

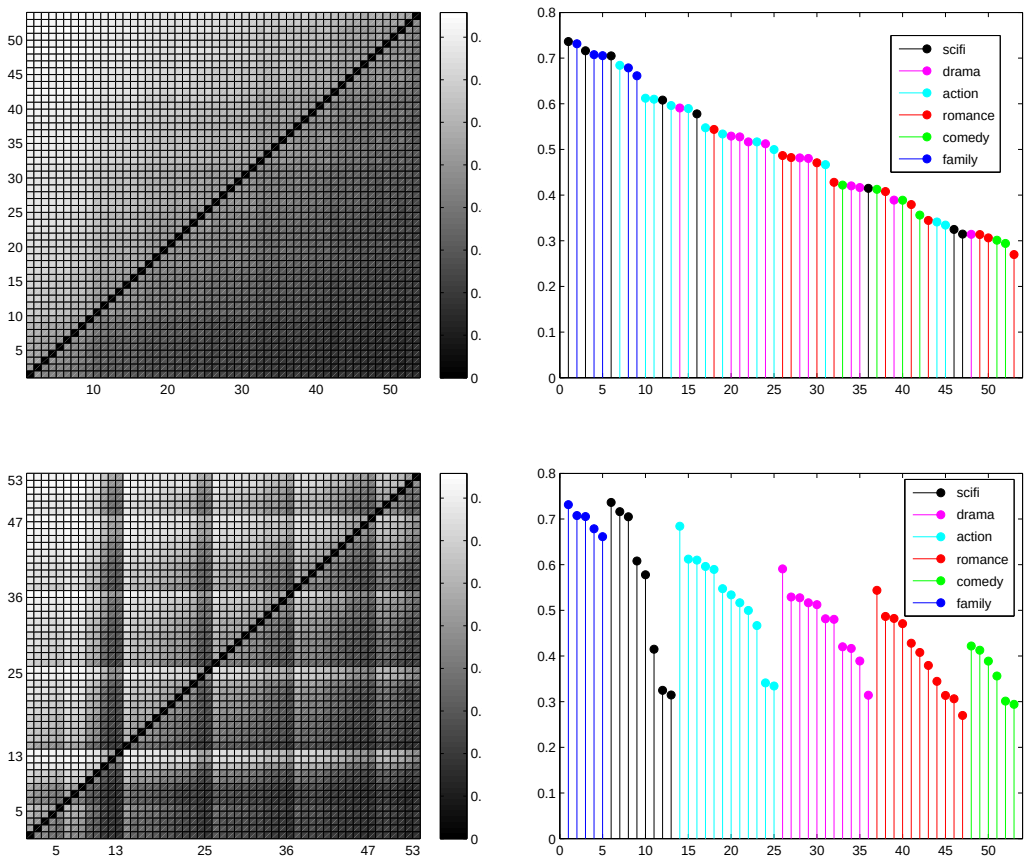


Figure 14: Left: The estimated probability of movie i being preferred to movie j. Right: a plot of $r(i) = \sum_j \hat{p}(i \prec j)/n$ for all movies with color indicating genres. In both panels the movies were ordered by $r(i)$ (top row) and first by popularity of genres and then by $r(i)$ (bottom row).

movies, on the other hand, enjoy high preference overall but exhibit a larger amount of variability as a few movies are among the least preferred. Similarly, the preference probabilities of action movies are widely spread with some movies being preferred to others and others being less preferred. More specifically (see bottom left panel of Figure 14) we see that the decay of $r(i)$ within genres is linear for family and romance and nonlinear for science fiction, action, drama, and comedy. In these last three genres there are a few really "bad" movies that are substantially lower than the rest of the curve. Table 8 shows the full information including titles, genres and orderings of the 53 most rated movies in Netflix.

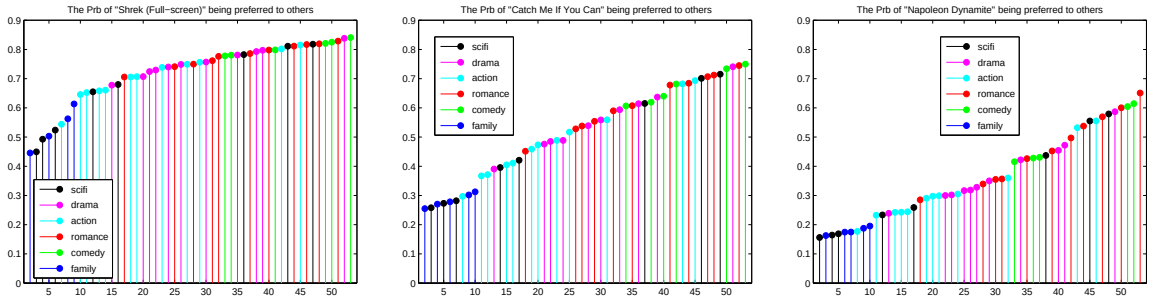


Figure 15: The value $\hat{p}(i \prec j)$ for all j for three movies: Shrek (left), Catch Me If You Can (middle) and Napoleon Dynamite (right).

We plot the individual values of $\hat{p}(i \prec j)$ for three movies: Shrek (family), Catch Me If You Can (drama) and Napoleon Dynamite (comedy) (Figure 15). Comparing the three stem plots we observe that Shrek is preferred to almost all other movies, Napoleon Dynamite is less preferred than most other movies, and Catch Me If You Can is preferred to some other movies but less preferred than others. Also interesting is the linear increase of the stem plots for Catch Me If You Can and Napoleon Dynamite and the non-linear increase of the stem plot for Shrek. This is likely a result of the fact that for very popular movies there are only a few comparable movies with the rest being very likely to be less preferred movies ($\hat{p}(i \prec j)$ close to 1).

In a second experiment (see Figure 16) we compare the predictive behavior of the kernel smoothing estimator with that of a parametric model (Mallows model) and the empirical measure (frequency of event occurring in the m samples). We select the 25 most rated movies for all three data sets and select a random sample of 1250 users for MovieLens, 10000 users for Netflix, and 3750 users for EachMovie. Each sampled data set is partitioned into a training set and a test set, containing 80% and 20% users, respectively. We construct the estimator based on the training set and evaluated the log-likelihood on the test set. We repeat the procedure three times and evaluate the average performance. A higher test set log-likelihood indicates that the model assigns high probability to events that occurred. Mathematically, this

Table 8: The table contains the information of the 53 most rated movies of Netflix. Columns are movie titles, genres, order1 (the ordering in the upper row of Figure 14), order2 (the ordering in the bottom row of Figure 14) and average ratings. Genres indicated by numbers from 1 to 6 represent science fiction, drama, action, romance, comedy, and family. The correlation between the average ratings and the average probabilities of being preferred to others $r(i) = \sum_j \hat{p}(i \prec j)/n$ is 0.93.

Movie	Genre	Order1	Order2	Mean
Finding Nemo	6	2	1	4.27
Shrek	6	4	2	4.07
The Incredibles	6	5	3	4.06
Monsters, Inc.	6	8	4	4.01
Shrek II	6	9	5	4.13
Lord of the Rings: the Return of the King	1	1	6	4.40
Lord of the Rings: the Two Towers	1	3	7	4.41
Lord of the Rings: the Fellowship of the Ring	1	6	8	4.42
Spider-Man II	1	12	9	3.99
Spider-Man	1	16	10	3.85
The Day After Tomorrow	1	36	11	3.43
Tomb Raider	1	46	12	2.86
Men in Black II	1	47	13	3.04
Pirates of the Caribbean I	3	7	14	4.29
The Last Samurai	3	10	15	3.94
Man on Fire	3	11	16	3.84
The Bourne Identity	3	13	17	3.99
The Bourne Supremacy	3	15	18	3.91
National Treasure	3	17	19	3.53
The Italian Job	3	19	20	3.75
Kill Bill II	3	23	21	3.47
Kill Bill I	3	25	22	3.60
Minority Report	3	31	23	3.61
S.W.A.T.	3	44	24	3.09
The Fast and the Furious	3	45	25	2.84
Ocean's Eleven	2	14	26	3.98
I, Robot	2	20	27	3.72
Mystic River	2	21	28	3.54
Troy	2	22	29	3.61
Catch Me If You Can	2	24	30	3.73
Big Fish	2	28	31	3.35
Collateral	2	29	32	3.60
John Q	2	34	33	3.07
Pearl Harbor	2	35	34	3.23
Swordfish	2	39	35	3.22
Lost in Translation	2	48	36	2.56
50 First Dates	4	18	37	3.76
My Big Fat Greek Wedding	4	26	38	3.60
Something's Gotta Give	4	27	39	3.43
The Terminal	4	30	40	3.47
How to Lose a Guy in 10 Days	4	32	41	3.33
Sweet Home Alabama	4	38	42	3.29
Sideways	4	41	43	2.54
Two Weeks Notice	4	43	44	3.11
Mr. Deeds	4	49	45	2.92
The Wedding Planner	4	50	46	2.71
Maid in Manhattan	4	53	47	2.52
The School of Rock	5	33	48	3.33
Bruce Almighty	5	37	49	3.51
Dodgeball: a True Underdog Story	5	40	50	3.19
Napoleon Dynamite	5	42	51	2.57
The Royal Tenenbaums	5	51	52	2.39
Anger Management	5	52	53	3.03

corresponds to approximating the KL divergence between nature and the model.

Since the Mallows model is intractable for large n we experimented with rankings over a small number of items. Specifically, we choose $n = 2, 3, 4$ items for MovieLens and $n = 3, 4, 5$ items for Netflix and EachMovie. For $n = 2$, we compute the log-likelihood restricted to two items i and j, both of which are chosen from the 25 most rated items. There are possible $\binom{25}{2}$ cases for each test data and the log-likelihood is averaged over all cases. The computation of the log-likelihood is similar for the case of $n = 3, 4, 5$.

Figure 16 shows the test set log-likelihood of different models as a function of training set size. For each model, the log-likelihood is displayed with the optimal parameters. For the parametric mallows model, the parameters are the location and the variance. The choices of location are all possible permutations of n items. For the variance, we experiment with a wide range of values from 0.001 to 10. The parameters are fitted by maximum likelihood based on the training data. For our kernel smoothing estimator, a moderate bandwidth h provides the best trade-off between fitting bias and model variance. In practice, we normalize the Kendall's tau distance for rankings of n items by dividing the maximum distance $n(n-1)/2$ and experiment the kernel bandwidth h with values ranging from 0 to 1 with step size 0.01. We choose the best value via 5-fold cross validation.

We observe that the kernel estimator consistently achieves higher test set log-likelihood than the Mallows model and the empirical measure. The former is due to the breakdown of parametric assumptions as indicated by Figure 2 (note that this happens even for n as low as 3). The latter is due to the superior statistical performance of the kernel estimator over the empirical measure.

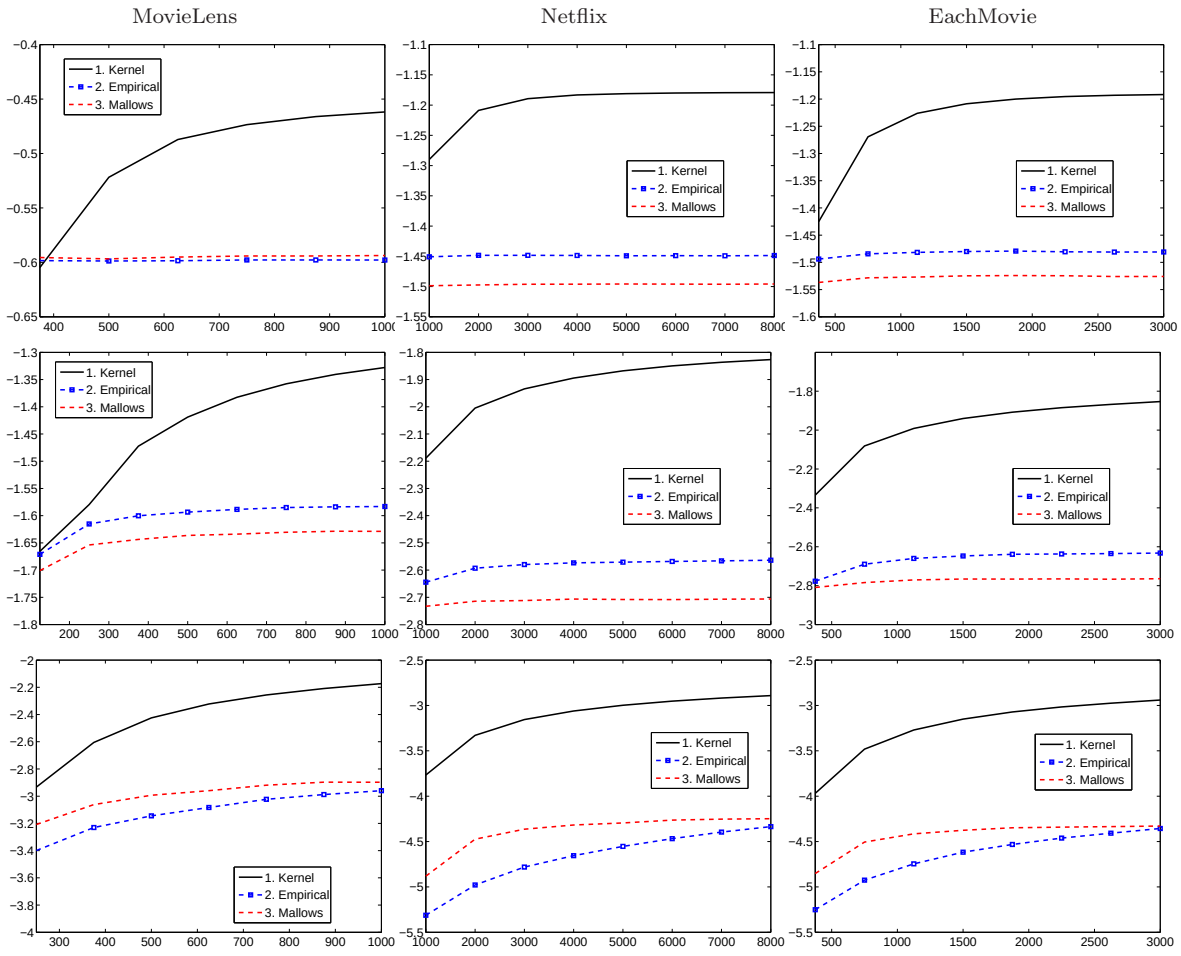


Figure 16: The test-set log-likelihood for kernel smoothing, Mallows model, and the empirical measure with respect to training size m (x-axis) for a small number of items n (top, middle, bottom rows) on three data sets. Specifically, $n = 3, 4, 5$ (top, middle, bottom rows) for the Netflix and EachMovie data sets and $n = 2, 3, 4$ (top, middle, bottom rows) for the MovieLens datset. Both of the Mallows model (which is also intractable for large n which is why $n \leq 5$ in the experiment) and the empirical measure perform worse than the kernel estimator $\hat{p}$.

4.4.2 Rank Prediction

Our task here is to predict ranking of new unseen items for users. We follow the standard procedure in collaborative filtering: the set of users is partitioned to two sets, a training set and a testing set. For each of the test set users we further split the observed items into two sets: one set used for estimating preferences (together with the preferences of the training set users) and the second set to evaluate the performance of the prediction [72]. We experiment with three different data sets. For MovieLens data set, we randomly partition the data to a training set containing 5000 users and a test set containing 1040 users. For Netflix data set, we select the 800 most rated movies and construct a random sampled training set containing 7000 users and a test set containing 3000 users. For EachMovie data set, we select the 1000 most rated movies and construct a random sampled training set containing 7000 users and a test set containing 3000 users. We further split each test set into two sets: observed test set and evaluation test set, containing 75% and 25% items respectively. We repeat the procedure three times and evaluate the average performance.

The performance of an algorithm is evaluated by a loss function $L(i, j)$, which measures the loss of predicting rank i when true rank is j (rank here refers to the number of sets of equivalent items that are more or less preferred than the current item). We focus on three loss functions: $L_0(i, j) = 0$ if $i = j$ and 1 otherwise, $L_1(i, j) = |i - j|$ which reduces to the standard CF evaluation technique described in [72], and an asymmetric loss function (rows correspond to estimated number of stars

(0-5) and columns to actual number of stars (0-5)

$$L_e = \begin{pmatrix} 0 & 0 & 0 & 3 & 4 & 5 \\ 0 & 0 & 0 & 2 & 3 & 4 \\ 0 & 0 & 0 & 1 & 2 & 3 \\ 9 & 4 & 1.5 & 0 & 0 & 0 \\ 12 & 6 & 3 & 0 & 0 & 0 \\ 15 & 8 & 4.5 & 0 & 0 & 0 \end{pmatrix}. \tag{34}$$

In contrast to the L_0 and L_1 loss, L_e captures the fact that recommending bad movies as good movies is worse than recommending good movies as bad.

For example, consider a test user whose observed preference is $3 \prec 4, 5, 6 \prec 10, 11, 12 \prec 23 \prec 40, 50, 60 \prec 100, 101$. We may withhold the preferences of items $4, 11$ for evaluation purposes. The recommendation systems then predict a rank of 1 for item 4 and a rank of 4 for item 11. Since the true ranking of these items are 2 and 3 the absolute value loss is $|1 - 2| = 1$ and $|3 - 4| = 1$ respectively.

In our algorithm, a probability density is estimated by applying the estimator to the training data and this density is then used to predict rankings for a test user conditional upon the test user's observed preferences. In particular, we use the kernel estimator $\hat{p}$ to predict ranks that minimize the posterior loss and thus adapts to customized loss functions such as L_e. The prediction in this case is a refinement $\delta(A)$ of the input ranking A which seeks to approximate the true preference B, i.e. the loss function $L(\delta(A), B)$ quantifies the adverse effect of recommending according to the rule $A \to \delta(A)$. Specifically, assuming an appropriate loss function is given we select the prediction rule δ that minimizes the posterior loss

$$\delta(A) = \underset{\delta(A) \in \delta^*(A)}{\arg\min} \; \mathsf{E}_{\hat{p}(B|A)} \, L(\delta(A), B) \tag{35}$$

where $\delta^*(A)$ is a set of potential refinements of A under consideration. This is an advantage of a probabilistic modeling approach over more ad-hoc rule based recommendation systems.

Figure 17 compares the performance of our estimator to several standard baselines in the collaborative filtering literature: two older memory based methods vector similarity (sim1), correlation (sim2) in [10], and a recent state-of-the-art non-negative matrix (NMF) factorization (gnmf) [56]. The kernel smoothing estimate performed similar to the state-of-the-art but substantially better than the memory based methods to which it is functionally similar.

We adopt the following parameter settings for the algorithms. For two memory based methods, a default voting is assigned to unrated items and we set the value to 3. For the matrix factorization method, we set the latent dimension to $k = 15$ and optimization iteration to 10 as recommended by [56]. For our algorithm, we apply 5-fold cross validation to determine the kernel bandwidth h. Figure 18 shows the kernel bandwidth selection via cross validation. Optimal performance is achieved with h ranging from 0.14 to 0.25 for EachMovie data set and 0.1 to 0.14 for Netflix data set.

4.4.3 Rule Discovery

In the third task, we construct the estimator for the $n = 100$ most rated movies in Netflix data set consisting of $m = 10000$ users who rate most of these movies. The kernel bandwidth of the estimator is set to $h = 0.29$. We use the estimator $\hat{p}$ to detect noteworthy association rules of the type $i \prec j \Rightarrow k \prec l$ (if i is preferred to j then it is probably the case that k is preferred to l). Such association rules are important for both business analytics (devising marketing and manufacturing strategies) and recommendation system engineering. Specifically, we use $\hat{p}$ to select sets of four items i, j, k, l for which the mutual information $I(i \prec j\,; k \prec l)$ is maximized. The mutual

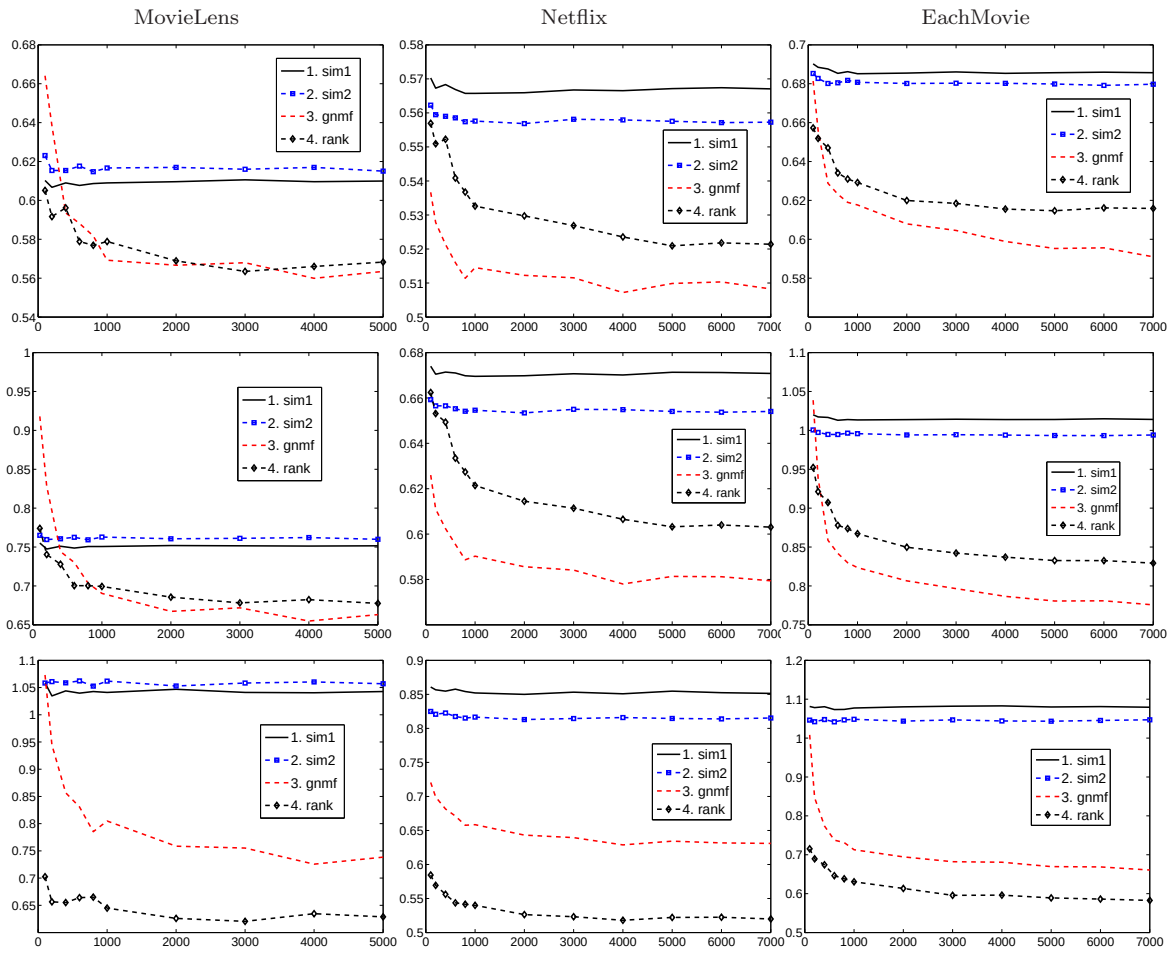


Figure 17: The prediction loss (top row: 0/1 loss L_0, middle row: L_1 loss, bottom row: asymmetric loss L_e) with respect to training size on three data sets (MovieLens 6040 users rating 3952 movies, Netflix 10000 users rating 800 movies, and EachMovie 10000 users rating 1000 movies). The kernel smoothing estimate performed similar to the state-of-the-art gnmf (matrix factorization) but substantially better than the memory based methods to which it is functionally similar.

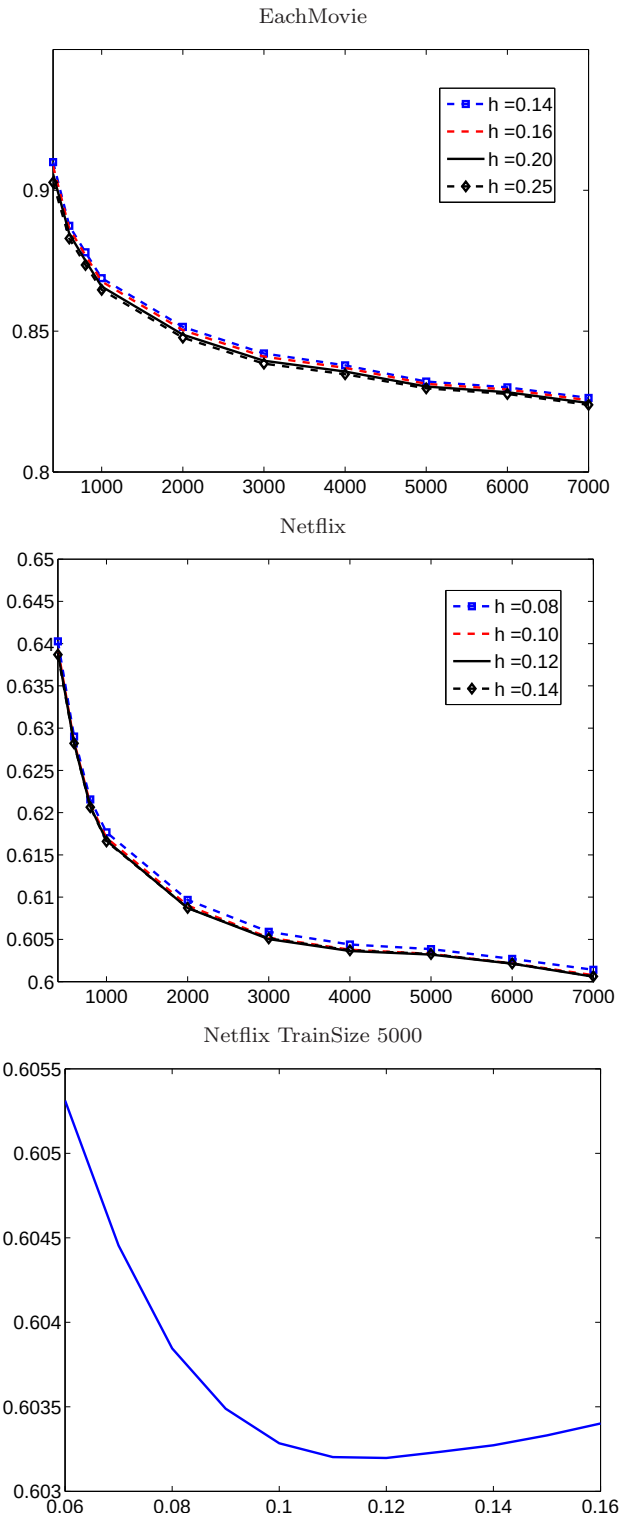


Figure 18: The L_1 loss using our estimator with different kernel bandwidths h with respect to training size on data sets EachMovie (10000 users and 1000 movies) and Netflix (10000 users and 800 movies). The x-axis is the training size. The figure on the bottom shows the loss using different kernel bandwidths when the training size is fixed. The x-axis is the kernel bandwidth.

information is

$$
\begin{aligned}
I(i \prec j\,; k \prec l) = p(i \prec j \cap k \prec l) \log \frac{p(i \prec j \cap k \prec l)}{p(i \prec j)p(k \prec l)} \\
+ p(j \prec i \cap k \prec l) \log \frac{p(j \prec i \cap k \prec l)}{p(j \prec i)p(k \prec l)} \\
+ p(i \prec j \cap l \prec k) \log \frac{p(i \prec j \cap l \prec k)}{p(i \prec j)p(l \prec k)} \\
+ p(j \prec i \cap l \prec k) \log \frac{p(j \prec i \cap l \prec k)}{p(j \prec i)p(l \prec k)}
\end{aligned}
\tag{36}
$$

After these sets are identified we detected the precise shape of the rule (i.e., $i \prec j \Rightarrow k \prec l$ rather than $j \prec i \Rightarrow k \prec l$ by examining the summands in the mutual information expectation).

Figure 19 (top) shows the top 10 rules that were discovered from $\binom{100}{4}$ rules in total. These rules nicely isolate viewer preferences for genres such as fantasy, romantic comedies, animation, and action (note however that genre information was not used in the rule discovery). To quantitatively evaluate the rule discovery process we judge a rule $i \prec j \Rightarrow k \prec l$ to be good if i, k are of the same genre and j, l are of the same genre. This quantitative evaluation appears in Figure 19 (bottom) where it is contrasted with the same rule discovery process (maximizing mutual information) based on the empirical measure. Although this rule discovery experiment is scored as a success based on the genre of the movies identified, this criteria is only a proxy for movie similarity in the absence of some known measurement. A qualitative examination of these results shows that much more than genre is recovered, e.g. figure 19 rule 1 states if "Shrek $\prec$ LOTR: The Fellowship of the Ring" then "Shrek 2 $\prec$ LOTR: The Return of the King", that is movies of the same series are identified as preferred to other movies in the same series. Figure 20 shows the top rules that were discovered within the same genre.

In another rule discovery experiment, we use $\hat{p}$ to detect association rules of the

Shrek $\prec$ LOTR: The Fellowship of the Ring	$\Rightarrow$	Shrek 2$\prec$ LOTR: The Return of the King
Shrek $\prec$ LOTR: The Fellowship of the Ring	$\Rightarrow$	Shrek 2$\prec$ LOTR: The Two Towers
Shrek 2 $\prec$ LOTR: The Fellowship of the Ring	$\Rightarrow$	Shrek$\prec$ LOTR: The Return of the King
Kill Bill 2 $\prec$ National Treasure	$\Rightarrow$	Kill Bill 1 $\prec$ I. Robot
Shrek 2 $\prec$ LOTR: The Fellowship of the Ring	$\Rightarrow$	Shrek 2$\prec$ LOTR: The Two Towers
LOTR: The Fellowship $\prec$ Monsters, Inc. of the Ring	$\Rightarrow$	LOTR: The Two Towers$\prec$ Shrek
National Treasure $\prec$ Kill Bill 2	$\Rightarrow$	Pearl Harbor $\prec$ Kill Bill 1
LOTR: The Fellowship $\prec$ Monsters, Inc. of the Ring	$\Rightarrow$	LOTR: The Return $\prec$ Shrek of the King
How to Lose a Guy in 10 Days $\prec$ Kill Bill 2	$\Rightarrow$	50 First Dates$\prec$ Kill Bill 1
I, Robot $\prec$ Kill Bill 2	$\Rightarrow$	The Day After Tomorrow $\prec$ Kill Bill 1

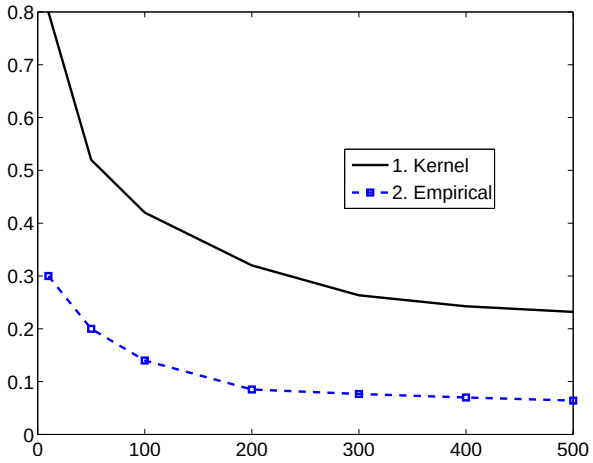


Figure 19: Top: top 10 rules discovered by the kernel smoothing estimator on Netflix in terms of maximizing mutual information. Bottom: a quantitative evaluation of the rule discovery. The x axis represents the number of rules discovered (i.e. increasing values on the x-axis correspond to selecting additional sets of movies with decreasing mutual information) and the y axis represents the frequency of good rules in the discovered rules. Here a rule $i \prec j \Rightarrow k \prec l$ is considered good if i, k are of the same genre and j, l are of the same genre.

Spider-Man $\prec$ LOTR: The Fellowship of the Ring	$\Rightarrow$	Spider-Man 2 $\prec$ LOTR: The Return of the King
The Day After Tomorrow$\prec$ LOTR: The Fellowship of the Ring	$\Rightarrow$	Spider-Man 2$\prec$ LOTR: The Return of the King
Men in Black II $\prec$ Spider-Man	$\Rightarrow$	Tomb Raider $\prec$ Spider-Man 2
Pearl Harbor $\prec$ Catch Me If You Can	$\Rightarrow$	Troy $\prec$ Mystic River
I, Robot $\prec$ Catch Me If You Can	$\Rightarrow$	Troy $\prec$ Ocean's Eleven
Collateral$\prec$ I, Robot	$\Rightarrow$	Lost in Translation$\prec$ Pearl Harbor
National Treasure $\prec$ Kill Bill 2	$\Rightarrow$	S.W.A.T.$\prec$ Kill Bill 1
The Fast and the Furious $\prec$ Kill Bill 2	$\Rightarrow$	The Italian Job $\prec$ Kill Bill 1
The Bourne Supremacy$\prec$ Man on Fire	$\Rightarrow$	The Bourne Identity$\prec$ The Last Samurai

Figure 20: Rules within the same genre (top 3 science fiction, middle 3 drama, and bottom 3 action) discovered by the kernel smoothing estimator on Netflix in terms of maximizing mutual information. The rules are in the form of $i \prec j \Rightarrow k \prec l$, where i, j, k, l are of the same genre.

form i ranked highest $\Rightarrow$ j ranked second highest by selecting i, j that maximize the score $\frac{p(\pi(i)=1,\pi(j)=2)}{p(\pi(i)=1)p(\pi(j)=2)}$ between pairs of movies in the Netflix data. We similarly detected rules of the form i ranked highest $\Rightarrow$ j ranked lowest by maximizing the scores $\frac{p(\pi(i)=1,\pi(j)=\text{last})}{p(\pi(i)=1)p(\pi(j)=\text{last})}$ between pairs of movies.

The upper panel of Table 9 shows the top 9 rules of 100 most rated movies, which nicely represents movie preference of similar type, e.g. romance, comedies, and action. The bottom of Table 9 shows the top 9 rules that represents like and dislike of different movie types, e.g. like of romance leads to dislike of action/thriller. While the paired movies in the left panel both come from the same genre, examining a specific pair "The Royal Tenenbaums $\Rightarrow$ American Beauty" and referring to the IMDB descriptions reveal that these movies are described by many common terms such as "Dark Humor, Depression, Deadpan, Drugs, Husband-Wife Relationship,...". To summarize, although the relationships identified are in part judged to be good or bad by genre, qualitatively these relationships can seen to be much closer than those obtained by randomly selecting movies from the same genre.

Table 9: Top rules discovered by kernel smoothing estimate on Netflix. Top: like A $\Rightarrow$ like B. Bottom: like A $\Rightarrow$ dislike B.

Kill Bill 1	$\Rightarrow$	Kill Bill 2
Maid in Manhattan	$\Rightarrow$	The Wedding Planner
Two Weeks Notice	$\Rightarrow$	Miss Congeniality
The Royal Tenenbaums	$\Rightarrow$	Lost in Translation
The Royal Tenenbaums	$\Rightarrow$	American Beauty
The Fast and the Furious	$\Rightarrow$	Gone in 60 Seconds
Spider-Man	$\Rightarrow$	Spider-Man 2
Anger Management	$\Rightarrow$	Bruce Almighty
Memento	$\Rightarrow$	Pulp Fiction
Maid in Manhattan	$\Rightarrow$	Pulp Fiction
Maid in Manhattan	$\Rightarrow$	Kill Bill: 1
How to Lose a Guy in 10 Days	$\Rightarrow$	Pulp Fiction
The Royal Tenenbaums	$\Rightarrow$	Pearl Harbor
The Wedding Planner	$\Rightarrow$	The Matrix
Peal Harbor	$\Rightarrow$	Memento
Lost in Translation	$\Rightarrow$	Pearl Harbor
The Day After Tomorrow	$\Rightarrow$	American Beauty
The Wedding Planner	$\Rightarrow$	Raiders of the Lost Ark

In a third experiment, we use $\hat{p}$ to construct an undirected graph where vertices are items (Netflix movies) and two nodes i,j are connected by an edge if the average score of the rule i ranked highest $\Rightarrow$ j ranked second highest and the rule j ranked highest $\Rightarrow$ i ranked second highest is higher than a certain threshold. Figure 21 shows the graph for the 100 most rated movies in Netflix (only movies with vertex degree greater than 0 are shown). The clusters in the graph corresponding to vertex color and numbering were obtained using a graph partitioning algorithm [42] and the graph is embedded in a 2-D plane using standard graph visualization technique [42]. Within each of the identified clusters movies are clearly similar with respect to genre, while an even finer separation can be observed when looking at specific clusters. For example, clusters 6 and 9 both contain comedy movies, where as cluster 6 tends toward slapstick humor and cluster 9 contains romantic comedies.

Cluster	Movie Titles
1	American Beauty, Lost in Translation, Pulp Fiction, Kill Bill I,II, Memento, The Royal Tenenbaums, Napoleon Dynamite,..
2	Spider-Man, Spider-Man II
3	Lord of the Rings I,II,III
4	The Bourne Identity, The Bourne Supremacy
5	Shrek, Shrek II
6	Meet the parents, American Pie
7	Indiana Jones and the Last Crusade, Raiders of the Lost Ark
8	The Patriot, Pearl Harbor, Men of Honor, John Q, The General's Daughter, National Treasure, Troy, The Italian Job,..
9	Miss Congeniality, Sweet Home Alabama, Two Weeks Notice, 50 First Dates, The Wedding Planner, Maid in Manhattan, Titanic,..
10	Men in Black I,II, Bruce Almighty, Anger Management, Mr. Deeds, Tomb Raider, The Fast and the Furious
11	Independence Day, Con Air, Twister, Armageddon, The Rock, Lethal Weapon 4, The Fugitive, Air Force One

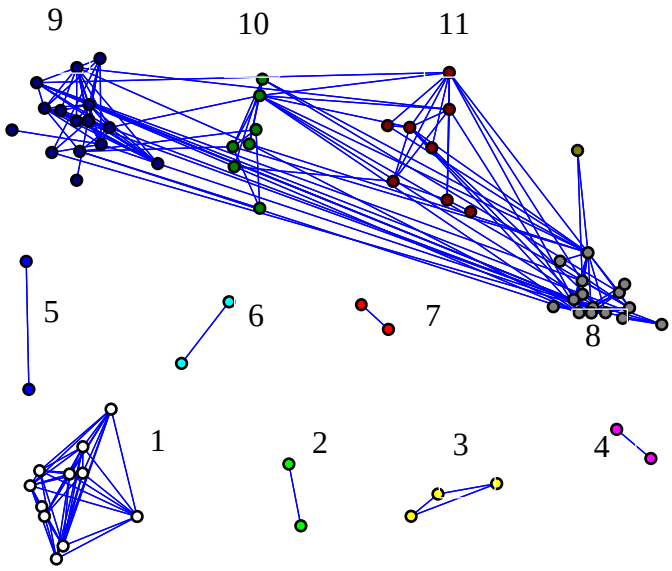


Figure 21: A graph corresponding to the 100 most rated Netflix movies where edges represent high affinity as determined by the rule discovery process (see text for more details).

CHAPTER V

PREFERENCE ELICITATION

Preference elicitation intend to solicit the interests of new users to the recommender system and rapidly generate a good initial set of recommendations. A popular strategy is asking users to rate certain items through an initial interview in order to learn user preferences. Users should be queried adaptively in a sequential fashion, and multiple items should be offered for opinion solicitation at each trial. In this work, we propose a novel algorithm that learns to conduct the interview process guided by a decision tree with multiple questions at each split. The splits, represented as sparse weight vectors, are learned through an L_1-constrained optimization framework. The users are directed to child nodes according to the inner product of their responses and the corresponding weight vector. More importantly, to account for the variety of responses coming to a node, a linear regressor is learned within each node using all the previously obtained answers as input to predict item ratings. A user study, preliminary but first in its kind in cold-start recommendation, is conducted to explore the efficient number and format of questions being asked in a recommendation survey to minimize user cognitive efforts. Quantitative experimental validations also show that the proposed algorithm outperforms state-of-the-art approaches in terms of both the prediction accuracy and user cognitive efforts.

5.1 Introduction

Recommender systems have been established as a critical tool for many business applications with significant economic impact. Successful systems span a variety of platforms, including Amazon's book recommendations, Netflix's movie recommendations, and Pandora's music recommendations. A popular and effective approach to

recommendations is collaborative filtering, which focuses on detecting users with similar preferences and recommending items favored by the like-minded [10, 54, 43, 72]. However, the system would fail to provide recommendations when no preference information is gathered from a new user, which is known as the cold-start problem [33, 76, 85].

Rapid profiling of new users is a key challenge in cold-start recommender systems. The most direct way is going through an initial interview process to solicit the preferences of new users [76]. In the interview, the recommender system asks users to provide opinions on selected items and constructs a rough user profile. Asking too few questions may lead to inaccurate estimation of user profiles and the system is unable to provide satisfactory recommendations, while asking too many questions may cause users to abandon the interview. A good interview is targeted at discovering user interests with minimum interactions. Specifically, the process should focus on (1) increasing recommendation accuracy and (2) minimizing user interaction efforts.

An adaptive interview process is known to improve accuracy, compared with previous approaches such as methods based on meta-features or a static set of interview questions. Many state-of-the-art work [76, 104, 33] have championed decision trees [11] as a natural fit for adaptive interviews of new users. In a primitive form of the decision tree, each node asks for user opinions on one item and users are directed to subtrees based on whether their answer is *like*, *dislike* or *unknown*. Finally, the average preference of training users within each leaf node is used to generate the recommendation list.

However, most users do not have the luxury to go through the entire item set, thus the system needs to serve a majority of users who have seen only tiny percentage of the items. In this case the constructed decision trees are often extremely unbalanced with the *unknown* branch capturing more than 80% of the users at each split (Figure 22, upper). Many users have to repeatedly select *unknown* multiple times before

locating any item they know about. In this case, not only did the system gain little valuable information on users' interests, but the users also easily get bored and may opt out of the system prematurely.

For a better cold-start recommender system, we advocate designs focusing on asking multiple questions at each trial. Displaying multiple questions on one screen increases the chance that a user knows at least one of them and thus allowing for solicitation of more valuable information. This was also suggested by some experimental studies that show users prefer providing opinions on multiple items at a trial [76]. However, within the collaborative filtering context, there has not been much existing work that focuses on building a single decision tree with each node asking multiple questions.

The major contribution of this work is to develop a tree learning algorithm for cold-start collaborative filtering with each node asking multiple questions. There are two key technical challenges to overcome in learning such a tree structure. First, with multiple questions at each node, it becomes substantially more expensive to search over all possible splits. Instead, we rely on a framework of minimizing expected prediction loss with L_1 regularization to find each split. Second, when asking multiple questions, users with different opinions would follow the same branch, and relying on average training user response within each node is no longer sufficient. We propose to learn a regressor within each node making use of all the answers from the root to the present node. In our experiments, the algorithm is shown to be able to improve the performance of the cold-start recommendation system. One example of the tree with multiple questions is shown in Figure 22 (lower).

In addition to prediction accuracy, another essential component for profiling new users is minimizing user efforts in the interaction. Critical features influencing user cognitive efforts include the number of items to ask at each trial, as well as the type of answers collected from the users. User response in common recommender systems

can be binary or 5-star rating scale. The 5-star rating response is expected to be more informative than a binary response but it may take more time in the user interaction. However, few existing work provides a quantitative analysis on the efforts users would spend on different interfaces.

This issue is addressed in our second contribution which quantitative analyzes different strategies in terms of both prediction accuracy and user efforts. A user study, preliminary but first in its kind, is performed to measure user efforts by the interaction time. Tree-models that ask different number of questions and different type of question (binary/5-scale) are shown randomly to different users through a web interview interface. The average time spent on each scenario is measured and we are thus able to measure accuracy against average user interaction time, which reveals the configurations that work best in real-life scenarios. The current quantitative user study that compares $1-4$ questions in both binary and 5-star formats seems to be new, to the best of our knowledge.

5.2 Related Work

There have been a substantial body of literature on collaborative filtering (CF) recommendations, which can largely be classified into memory and model based methods. Memory-based CF methods predict the ratings of items based on the similarity between the test user and training users [79, 10, 40]. Similarity measures include Pearson correlation [79] and Vector cosine similarity [10, 40]. Other memory-based CF methods include item-based CF [84] and a non-parametric probabilistic method based on ranking preference similarities [92]. Model-based CF includes user and item clustering, Bayesian networks [10], and probabilistic latent variable models [72, 103, 43]. The state-of-the-art methods, including the Netflix competition winner, are mostly based on matrix factorization. The factorized matrix can be used to fill out the unobserved entries of the user-rating matrix in a way similar to latent factor analysis [78, 54].

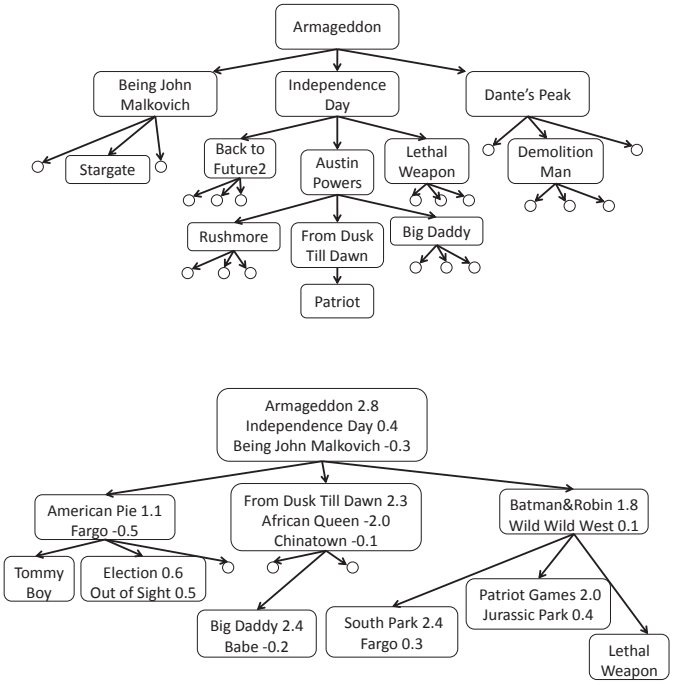


Figure 22: The interview trees learned from the MovieLens data set. A circle denotes a leaf node with recommendation results. Top: each node asks a single question. The first 5 levels are shown from the 10-level learned tree. Users are branched to left, middle and right subtrees according to the answers *like*, *unknown* and *dislike* respectively. It can be seen that the middle branch of the tree is much longer than the other branches. Bottom: each node asks multiple questions. The first 3 levels out of a total 5 are shown. Users are branched to left, middle and right subtrees according to equation (39) given the answers and corresponding weights.

One difficulty in recommender system is that the performance is affected by factors such as number of users, number of items, and observed total ratings. Unified models have been proposed in [1, 36, 85] by combining both CF methods and content features such as the genre, actor and director of a movie. These methods generally achieve better prediction accuracy on the items that few or no users have ever rated.

Several studies focus on eliciting preference for new users to solve the cold-start problem. It has been demonstrated in [74] that a good elicitation strategy should increase prediction accuracy with minimum user interaction. Several approaches construct a short interview in which users are asked to provide information on selected

items. Earlier attempts [76, 77] construct a static seed set based on the item popularity and informativeness. A later work [32] proposes a greedy algorithm to select the seed set that minimizes the prediction error. The performance of seed-based methods is unsatisfactory because items selected in batch are not adaptive to user responses in the interview process. Active learning methods for collaborative filtering [9, 37, 47, 38] select questions according to criteria such as minimizing the expected distance to the true user model, although these methods are not efficient enough for online user interview.

The IGCN algorithm proposed in [77] uses pre-defined user clusters to adaptively select questions in the interview. An alternative approach using decision trees is also mentioned in [77], where each node is a question and users are directed to subtrees based on their responses. The idea was later developed in [33] as a more disciplined approach that fits the decision tree to user ratings. To improve the prediction accuracy with missing data, the work [104] further integrates the decision tree with matrix factorization to fit the user ratings with a latent factor analysis. The limitation is that only one item is selected to ask at each node of the decision tree. It is highly likely that the user does not know any items in the first several interactions and the estimation would be inaccurate. One alternative suggested in [33] is selecting multiple items to ask at each interaction by bootstrapping multiple trees. The method selects from randomly generated trees and chooses a linear combination of which fits the training data best. However, this tends to ask more questions compared to building a single tree with multiple questions at each split, because each of the shallow trees would more likely be full and in that case, the number of questions increases linearly with the number of trees.

In this work, we propose a tree learning algorithm for collaborative filtering with each node asking multiple questions. In contrast to the work [33], our algorithm enables optimization-based estimation procedures for not only the prediction of user

profiling but also the selection of questions. Our algorithm achieves significant improvements in terms of prediction accuracy and user interaction time. Furthermore, it is able to handle heterogeneous preference information including binary and 5-star rating responses.

Existing user study in [76, 77] is measuring the user efforts by the number of pages they views before they rate a minimum number of items for the initial interview. Our user study seeks to discover both the efficient number of questions on each screen, and whether binary or 5-star is more effective in discovering user preferences.

5.3 Decision Tree for Cold-start Recommendation

In cold-start recommendations, we assume nothing about new users and an interview process is performed to ask the user a set of questions. Following the work in [33, 104], we ask questions in the form "Do you like the movie *50 first dates*?" and the answer could be in $\{-1, 0, 1\}$ corresponding to *like*, *dislike* and *unknown*, or a 5-scale rating, etc. Based on the answers, the system would adaptively ask another set of questions, gradually refine user preference and recommend a set of movies in the end. This fits well with a decision tree structure. Each node corresponds to one or multiple questions to ask. According to the evaluation of a user's answers, the user will be directed to subtrees. Once the user reaches the leaf node, the recommendation list is provided using the user preference learned from training data.

Let r_{ij} denote the observed rating of training user i for item j, where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. The pairs (i, j) are stored in the set $O = \{(i, j) \mid r_{ij} \text{ is observed}\}$. Denote $O_i = \{j \mid r_{ij} \text{ is observed}\}$ the set of items user i rated. Similarly, denote $O^j = \{i \mid r_{ij} \text{ is observed}\}$ the set of users that rate item j. Given a user i with the answer x_i, our goal is to learn a user profile $T(x_i)$ that predicts the rating of this user on all the items.

In the simplest form, the user profile $T(x_i)$ is an n dimensional vector with each

dimension representing the predicted rating of item j. However, such a representation is computationally challenging and hard to learn in large corpora with thousands or millions of items. A common technique in recommender systems is to obtain lower dimensional models by minimizing a prediction loss function between the predicted ratings and true ratings of users. For example,

$$\min_{T,V} \sum_{i} \sum_{j \in O_i} (r_{ij} - T(x_i)^\top v_j)^2, \tag{37}$$

where $V = [v_1, v_2, \ldots, v_n]$ is an $k \times n$ matrix and $T(x_i)$ is a k-dimensional vector. v_j can be viewed as a k-dimensional profile for item j. Usually, T and V are optimized using matrix factorization methods, such as PCA, NMF or nuclear-norm based matrix completion.

In our case, a decision tree is constructed from training user ratings in a top-down approach. $T(x)$ should be viewed as a function induced from the decision tree. For each node N in the tree, we learn a distinct function $T_N(x)$ for that node. During test time, a user is first assigned to a node based on his/her answers to the interview, then the function in that node is used to obtain his/her user profile.

We adopt the framework in (37) for V and will discuss the details for obtaining V in Section 5.3.4. However, our main contribution lies in determining the decision tree split with multiple questions, as well as learning the user profile function in each node. Therefore we discuss these first in Section 5.3.1 - 5.3.3.

5.3.1 Determining a Split with Multiple Questions

In this subsection, we propose our approach of learning to ask multiple questions at each decision tree split. For this goal, there are two challenges to overcome. The first is that searching over all possible splits becomes a combinatorial problem with a prohibitive $\binom{n}{l}$ possibilities to evaluate. We solve this problem by relaxing it as an L_1 regularized optimization. The second is that since we are keeping the structure of the tree with 3 branches, it is possible that users with different opinions would

enter the same node. Therefore different from other decision tree approaches (e.g. [104]), a non-constant function T needs to be learned separately for each node. We solve this second problem by training a linear regressor within each node using all the previously obtained answers as input.

For splitting with multiple items, the idea is to optimize for a sparse weight vector w on each item that has only a few non-zeros. Depending on whether $w^\top x$ is positive or negative, users are assigned into different subtrees. A third node corresponding to *unknown* is created by collecting the users with $w^\top x$ very close to 0.

Formally, let n-dimensional vector w denote the weight of items, and x_i denote the answer of user i to the items. Training users at the current node are split into 3 child nodes L, D and U according to the linear combination of the answers $x_i^\top w$. We use a modified logistic probability model to determine the split. Let p_i, q_i denote the probability that user i belongs to the L, and D branch respectively:

$$\begin{aligned} p_i &= \frac{1}{1 + c\exp(-x_i^\top w)}, \\ q_i &= \frac{1}{1 + c\exp(x_i^\top w)}. \end{aligned} \tag{38}$$

The three subgroups are defined as:

$$\begin{aligned} L(w) &= \{i | p_i > q_i,\ p_i > 1 - p_i - q_i\}, \\ D(w) &= \{i | q_i > p_i,\ q_i > 1 - p_i - q_i\}, \\ U(w) &= \{i | 1 - p_i - q_i \geq \max(p_i, q_i)\}, \end{aligned} \tag{39}$$

where c is a parameter controlling the likelihood of falling into different groups. In practice, we find that $c = 2$ works best. In such a case, a user belongs to the L group when $x_i^\top w$ is positive, the D group when $x_i^\top w$ is negative, and the middle group U only when the user answers none of the questions, as shown in Figure 23.

Ideally, we need to optimize such an objective function

$$\begin{aligned}\min_{w,T_L,T_D,T_U} & \sum_{i\in L(w)}\sum_{j\in O_i}(r_{ij}-T_L(x_i)^\top v_j)^2 \\ & + \sum_{i\in D(w)}\sum_{j\in O_i}(r_{ij}-T_D(x_i)^\top v_j)^2 \\ & + \sum_{i\in U(w)}\sum_{j\in O_i}(r_{ij}-T_U(x_i)^\top v_j)^2 \\ \text{s.t. } & \|w\|_0 \le l, \end{aligned} \tag{40}$$

where T_L, T_D and T_U are the profile functions for the nodes L, D, U, and $\|w\|_0$ denotes the number of non-zeros in w. The objective function first divides the training users into three child nodes, and then computes the squared error within each child node. The goal is to minimize the sum of errors in all the three child nodes. In addition, the constraint $\|w\|_0 \le l$ determined that w cannot have more than l non-zeros.

We would adopt an alternating minimization strategy that optimizes w and T_L, T_D, T_U iteratively. However, the optimization of w is a complicated non-convex combinatorial problem, therefore we make relaxations to solve it with continuous optimization. In the next two subsections we discuss separately the optimization of w and the computation of T_L, T_D, and T_U.

5.3.2 Relaxations for the Optimization of w

In order to solve this problem with continuous optimization, we adopt two relaxations. The first is to relax the hard partitioning L, D, and U into a soft partitioning, with the probability of x belonging to the subgroups L, D and U to be p_i, q_i, and $1-p_i-q_i$, respectively. This makes the main objective function smooth in w.

The second relaxation is to append a penalty term on $\|w\|_1$ to the objective function, instead of a hard constraint on the number of nonzeros $\|w\|_0$. This L_1 relaxation approach has been popular in machine learning and signal processing in recent years. The L_1 term is convex and there exist efficient methods to optimize a

smooth objective function with such a penalty term [86].

Let $\bar{m}$ denote the number of users reaching the current node, our relaxation problem computes w which minimizes the weighted prediction loss:

$$\begin{aligned}
&\min_{w} \sum_{i=1}^{\bar{m}} p_i \sum_{j \in O_i} (r_{ij} - T_L(x_i)^\top v_j)^2 \\
&+ \sum_{i=1}^{\bar{m}} q_i \sum_{j \in O_i} (r_{ij} - T_D(x_i)^\top v_j)^2 \\
&+ \sum_{i=1}^{\bar{m}} (1 - p_i - q_i) \sum_{j \in O_i} (r_{ij} - T_U(x_i)^\top v_j)^2 + \lambda ||w||_1. \qquad (41)
\end{aligned}$$

To solve (41), we adopt a projected scaled sub-gradient optimization [86]. This algorithm computes the Hessian only for the nonzero part of the current w, and adopts linear-time Barzilai-Borwein subgradient steps for the rest. It is suitable for our task because of its fast convergence rate thanks to the incorporated second-order information, and each iteration is a linear-time operation. The only cubic-time computation is to compute the Hessian inverse on the nonzeros. Since we have usually less than 5 nonzeros, this step would normally not take more than 5^3.

By changing the parameter λ, we can find solutions with different numbers of nonzeros in w. In practice, we first binary search between 0 and λ_{max} to find λ_0 such that the number of nonzero entries of w is between 1 and l. Then we search around λ_0 with a finer step size, locating l different solutions with $1, 2, \ldots, l$ nonzeros respectively. From this solution set, we select the one that minimize (41) without the penalty term.

The optimization for w is still non-convex, therefore a good choice of starting point is important. We choose the starting point as the best 1-item question found by the approach in [104] and set $w_s = 1$ and $w_j = 0$ $(j = 1, 2, \ldots, n, j \neq s)$ as our initial value. We find this scheme to work well in practice.

5.3.3 Computing the Profile Functions

We adopt a linear model in each node in order to allow different user preferences. The input to the linear model would be all the previous answers that the user has submitted. Therefore, the nodes deeper down the tree would have more information since more answers have been submitted by the user to arrive at the node.

Formally, let t ($t < n$) denote the number of asked items from the root till the current node, $\bar{x}_i$ denote the answer of user i, which is a $t+1$ dimensional vector including a constant dimension $\bar{x}_{i0} = 1$. Our linear model is $T_L(x_i) = Z_L\bar{x}_i$, $T_D(x_i) = Z_D\bar{x}_i$ and $T_U(x_i) = Z_U\bar{x}_i$, where Z_L, Z_D and Z_U are $(t+1) \times k$ matrices for each individual node in the next level.

We try to best approximate all the observed scores r_{ij} within the node, using all the obtained answers $\bar{x}_i$ and the current item profiles v_j. This gives the following optimization problem:

$$
\begin{aligned}
Z_L &= \arg\min_Z \sum_{i \in L(w)} \sum_{j \in O_i} (r_{ij} - (Z^\top \bar{x}_i)^\top v_j)^2, \\
Z_D &= \arg\min_Z \sum_{i \in D(w)} \sum_{j \in O_i} (r_{ij} - (Z^\top \bar{x}_i)^\top v_j)^2, \\
Z_U &= \arg\min_Z \sum_{i \in U(w)} \sum_{j \in O_i} (r_{ij} - (Z^\top \bar{x}_i)^\top v_j)^2.
\end{aligned} \tag{42}
$$

It turns out that each has a closed form solution:

$$
\begin{aligned}
z_L &= \left(\sum_{i \in L(w)} \sum_{j \in O_i} (\bar{x}_i \bar{x}_i^\top) \otimes (v_j v_j^\top) \right)^{-1} \left(\sum_{i \in L(w)} \sum_{j \in O_i} r_{ij} (\bar{x}_i \otimes v_j) \right), \\
z_D &= \left(\sum_{i \in D(w)} \sum_{j \in O_i} (\bar{x}_i \bar{x}_i^\top) \otimes (v_j v_j^\top) \right)^{-1} \left(\sum_{i \in D(w)} \sum_{j \in O_i} r_{ij} (\bar{x}_i \otimes v_j) \right), \\
z_U &= \left(\sum_{i \in U(w)} \sum_{j \in O_i} (\bar{x}_i \bar{x}_i^\top) \otimes (v_j v_j^\top) \right)^{-1} \left(\sum_{i \in U(w)} \sum_{j \in O_i} r_{ij} (\bar{x}_i \otimes v_j) \right),
\end{aligned} \tag{43}
$$

where z_L, z_D and z_U are large column vectors formed by concatenating the column vectors of the matrices Z_L, Z_D, and Z_U respectively, and $\otimes$ denotes the matrix

Kronecker product. The regression inside each node allows different answers to be mapped to different ratings, thus solving the problem of contradicting opinions for different users within a node. At each node, we alternatively optimize (41) and (42) until convergence.

When the amount of training data becomes small along the path reaching the leaf node, the estimation of user profiles may overfit. Following previous papers [104], we apply hierarchical regularization at the current node so that the coefficients of profile Z_L, Z_D and Z_U are shrinking towards the ones at its parent node. For example, the solution with regularization for Z_L is:

$$Z_L = \arg\min_{Z} \sum_{i \in L(w)} \sum_{j \in O_i} (r_{ij} - (Z^\top \bar{x}_i)^\top v_j)^2 + \lambda_z ||Z - Z_0||^2, \tag{44}$$

where Z_0 is the estimation at the parent node padded with zeros to reach the size of Z at the child node.

5.3.4 The Item Profile

The item profile is initialized using a nonlinear matrix factorization method based on Gaussian process latent variable models [56]. Given an initilized profiles v_j, we can construct a tree T using the algorithm in section 5.3.1. Given the decision tree T, the user profile estimated at leaf nodes of the tree is $T(x_i) = (Z_k^\top x_i)$, where x_i is the user responses along the path. We then update item profiles v_j $(j = 1, 2, \ldots, n)$ with regularized least square regression:

$$v_j = \arg\min_{v_j} \sum_{i \in O^j} (r_{ij} - v_j^\top T(x_i))^2 + \lambda_v ||v_j||^2. \tag{45}$$

A closed form solution for v_j exists as shown in [104].

5.3.5 Computational Complexity

The full algorithm is summarized in Algorithms 3 and 4. The computational complexity for updating item profiles v_j for all $j = 1, 2, \ldots, n$ is $O(n|O^j|_{max}k^2 + nk^3)$, where n is the number of items, $|O^j|$ is the number of users who rated item j, and k is the dimension of latent space. For the tree construction, at each node, the complexity for running the split algorithm in [104] is $O(\sum_{i \in users} |O_i|^2 + nk^3)$, the complexity for optimizing w and Z by (41) is $O(\alpha \sum_{i \in users} |O_i|n + \alpha t^3 k^3)$, where $|O_i|$ is the number of observed ratings of user i at the node, α is the iterations, and t is the maximum number of questions asked in the path. The complexity for building the whole tree is thus $O(d \sum_{i=1}^{m} |O_i|n + \beta nk^3 + \beta t^3 k^3)$, where d is the depth of the tree and β is the number of nodes in the tree. In practice, the tree depth d is no more than 6, the maximum number of questions t is no more than 20, and k is usually selected from 10 to 20.

Algorithm 3 Optimization for Tree Model and Item Profiles

Require: The training data $R = r_{ij}|(i, j) \in O$.
Ensure: Estimated decision tree T and item profile v_j $(j = 1, 2, \ldots, n)$.
1: Initialize v_j $(j = 1, 2, \ldots, n)$ using [56].
2: **while** not converge **do**
3: Fit a decision tree T using Algorithm 4.
4: Update v_j using Equation (45).
5: **end while**
6: **return** T and v_j $(j = 1, 2, \ldots, n)$.

5.4 Experiments

We examine the estimation framework on three movie recommendation data sets: MovieLens, EachMovie and Netflix. The details of each data set are shown in Table 10. The Netflix data we used is a random sample containing about half of the original movies and a quarter of the original users. The prediction performance is evaluated

Algorithm 4 Optimization for Tree Model

function FitTree(UsersAtNode)
Find the best single item s using [104].
Initialize w with all zeros but $w_s = 1$.
while not converge **do**
    Compute Z_L, Z_D, and Z_U using Equation (43).
    Update w using Equation (41).
end while
Split users into three groups $L(w)$, $D(w)$ and $U(w)$.
if square error reduces after split **and** depth ¡ maxDepth **then**
    call FitTree($L(w)$), FitTree($D(w)$) and FitTree($U(w)$) to construct subtrees.
end if
return T with $T(x) = (Z_k^\top x)^\top V$, if x falls in the k-th node in the decision tree.
end function

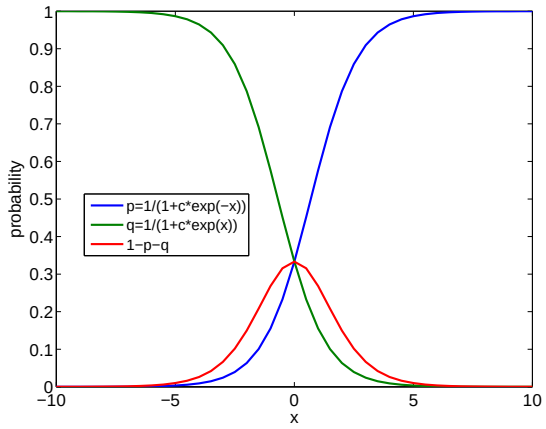


Figure 23: Probabilities of being in the three groups with $c = 2$.

on the test set S with the root mean square error (RMSE), which is defined as:

$$RMSE = \sqrt{\frac{1}{|S|} \sum_{(i,j) \in S} (\hat{r}_{i,j} - r_{i,j})^2}, \tag{46}$$

where $\hat{r}_{i,j}$ and $r_{i,j}$ are the predicted and ground truth ratings for user i and item j, respectively.

We seek to answer three questions in the following:

1. For new users, how does the proposed algorithm perform in terms of prediction accuracy comparing to baselines? How does the performance improve with respect to number of screens and questions?

2. How many questions do we ask on each screen in order to minimize user

interaction time?

3. What types of user responses are more informative and less time-consuming?

In the end, we summarize and analyze insights on how many items to query on each screen, what types of answers users can employ, and how prediction accuracy changes with the different settings.

Table 10: Data set Description

Data set	Users	Items	Ratings	Scale
MovieLens	6040	3952	$1,000,209$	$1-5$
EachMovie	$72,916$	1628	$2,811,983$	$1-6$
Netflix	$120,000$	8000	$11,670,175$	$1-5$

5.4.1 Cold-Start Prediction Accuracy

In this experiment, we randomly split the data into a training and a test set, each containing 75% and 25% users, respectively. All the ratings in the training set are used to construct the decision tree. The test set is further splited into two disjoint set: answer and evaluation sets, which contain 75% and 25% rated items for each test user. The answer set is used to simulate the responses of test users in the interview process. The evaluation set is used to evaluate the prediction accuracy after the interview process. The question is in the form "Do you like item j?". For example in the movie recommendation, the question is "Do you like movie *50 first date*?" For binary answer type, where a user is expected to answer *like*, *dislike* and *unknown*, we follow the standard settings [77, 33, 104] and simulate test user responses as the following:

$$x_{ij} = \begin{cases} 1 & (i,j) \in O \text{ and } r_{ij} > 3 \\ -1 & (i,j) \in O \text{ and } r_{ij} <= 3 \\ 0 & (i,j) \in O. \end{cases} \tag{47}$$

For the EachMovie data set where the rating scale is $1-6$, we use $r_{ij} > 4$ instead.

Figure 24 compares the performance of our model with several standard baselines. One is the single-question decision tree with matrix factorization, denoted as $w1_{Base}$ [104]. This model has been shown in [104] as the strongest tree model with single-question splits. The other is bootstrapping tree model $w4_{Base}$ [33] which generates random decision trees, and select l-best ones to form a linear combination. In our experiments, the tree generation method in [33] always performs worse than the one in [104]. Therefore in $w4_{Base}$ we adopted the bootstrapping methodology in [33] with the tree generation method from [104], in order for a fairer comparison. It has been shown that $1 \leq l \leq 5$ is practical for online interview. We set $l = 4$. Our models $w2$, $w3$, and $w4$ are the trees with maximum number of questions at each node being 2, 3, and 4 respectively.

We adopt the following parameter settings for the algorithms. For the algorithm $w1_{Base}$, the regularization parameters are set to $\lambda = 0.01$ and $\lambda_h = 0.03$ as recommended in [104]. For the algorithm $w4_{Base}$, we generalized 200 randomized trees with parameter $a = 10$ as reported in [33]. For our algorithm, the regularization parameters λ_z and λ_v are selected by 5-fold cross-validation on the training set. Usually the value of regularization parameter λ_z is moderate ranging from 0.03 to 0.1, which reduces the risk of overfitting. The dimensionality of the profiles for the matrix factorization model k is fixed to be 15. The optimization converges very quickly, usually within 10 iterations.

For all methods, the prediction error measured in RMSE decreases as more questions have been asked (Figure 24). In the first row of Figure 24, our models $w2, w3$ and $w4$ have a big advantage over $w1_{Base}$ when the error is measured per screen, since in each screen we have definitely solicited more information than $w1_{Base}$. The model $w4_{Base}$ also performs better than $w1_{Base}$ in this sense. But more notably, our models $w3$ and $w4$ outperform $w4_{Base}$, which shows the advantage of our multiple-question tree model versus a linear combination of multiple bootstrapped trees.

In the second row in Figure 24 where we compare performance versus number of questions, we see no major difference between $w1_{Base}$ and $w2, w3, w4$ when relatively few questions have been asked. However, our multiple-question tree model is able to ask more questions and further reduce the prediction error, whereas in $w1_{Base}$, the performance cannot improve with trees more than $6-7$ layers, because at this time the number of training users in each leaf node is already very small, essentially making further splits very susceptible to overfitting. The model $w4_{Base}$ in this row showed its inefficiency in needing to ask more questions to obtain the same performance, which comes from the redundancy and inefficient use of information from the multiple trees.

In the third row, we compare performance versus the expected time that users spend in the interview process measured from our user study, described in the next subsection. In this case, $w1_{Base}$ is comparable with our models when relatively few questions have been asked. However, our improvement is more significant if the user is going to spend more than 20 seconds in the recommender system. In that case the performance of single-question decision trees starts to saturate, but our models continue to provide more and more accurate recommendations to the user.

Table 11 shows an example of a user's responses and the recommendation list.

5.4.2 User Study

In this user study, we measure the expected time that users spend on the interview process. We created an online interview. The interview script loads the precomputed tree model and displays questions of the node on one screen. It proceeds to another screen after a user submits answers. The interview process ends when the user reaches a leaf node of the tree model. The trees are always downloaded fully into the local computer in order to save network communication time. There are 8 precomputed tree models. Let $w1_{Binary}$, $w2_{Binary}$, $w3_{Binary}$ and $w4_{Binary}$ denote the models with each node asking 1, 2, 3, and 4 questions maximum. The user responses for the

Table 11: A user's responses to the first two screens of the interview process each with 4 questions. The recommendation list after the interview is given below.

Interview questions

Screen	Questions	Responses
1	The Royal Tenenbaums	Unknown
	Lost in Translation	Like
	Independence Day	Unknown
	Being John Malkovich	Unknown
2	How to Lose a Guy in 10 Days	Like
	Miss Congeniality	Like
	Pulp Fiction	DisLike
	Taxi Driver	Unknown

Top-5 recommendations

Rank	Movie Title
1	Indiana Jones and the Last Crusade
2	The Sixth Sense
3	The Green Mile
4	Life Is Beautiful
5	Toy Story

tree models are binary *dislike*, *like*, or *unknown*. For the latter 4 tree models $w1_{Rate}$, $w2_{Rate}$, $w3_{Rate}$, and $w4_{Rate}$, user responses are in $1-5$ rating scale. Once a user logs into the interview, a random tree type will be selected. The users are instructed to answer the questions at their first sight, in order to avoid the situation that users linger on a particular page too long, or even search the internet to find particular answers.

In total we are able to collect responses from 76 users. Figure 25 shows our interview web interface. Table 12 compares the expected time to answer k questions on one screen. It reveals that the time increment for users to answer more questions per screen is usually sublinear, with answering 2 questions around 1.5 times slower than asking 1 question, and answering 4 questions around 2.5 times slower. Answering rating scale questions usually adds $2.5-3.5$ seconds to all the 4 settings ($1-4$ questions). The time difference between a rating scale answer and a binary answer seems not to have strong dependency with the number of questions. We suspect this

phenomenon still comes from the sparsity of the data: with 4 questions per page, usually the user would not have seen all movies on a page therefore he/she only has to provide ratings for $1-2$ of them.

Figure 26 compares the time of the interview process by asking different number of questions per screen. It can be seen that $w1_{Rate}$ does not seem to be a much better option than $w1_{Binary}$ because the time increased significantly without too many questions being asked. However, the fraction of time increased from $w4_{Binary}$ to $w4_{Rate}$ seems not as big. And although $w4_{Rate}$ costs the longest time (over 50 seconds for 4 screens), it is able to obtain about 14 rated responses, comparing to less than 5 rated responses from $w1_{Rate}$ in 35 seconds.

We plot in Figure 27 the prediction performance for these different settings. The performance differs with data, in MovieLens, the setting $w4_{Binary}$ with asking 4 questions for binary answers performed the best. In the Netflix data, the main advantage of a rated response is that it can achieve superior recommendation performance, with longer surveys. However, time-wise it has only a very small advantage if the total time spent on the survey is less than 30 seconds. In the EachMovie data however, a rated response is always significantly better than a binary one in all accounts. This study can provide material to support decisions in deploying cold-start recommender systems: if the goal is a short survey, then using either binary or rated responses should have no major difference, while rated responses could have a slight advantage. However if the goal is a longer survey for more accurate user responses, it is more likely that seeking rated responses from the user would perform better.

Table 12: A comparison of average time to answer n questions on one screen. When the number of questions per screen increases, the increase in the response time is sublinear.

Binary User Response

Ask l questions	$l = 1$	$l = 2$	$l = 3$	$l = 4$
Time (seconds)	4.4097	6.7267	7.8841	10.4232

Rating Scale User Response

Ask l questions	$l = 1$	$l = 2$	$l = 3$	$l = 4$
Time (seconds)	6.9339	9.5906	10.1404	13.8569

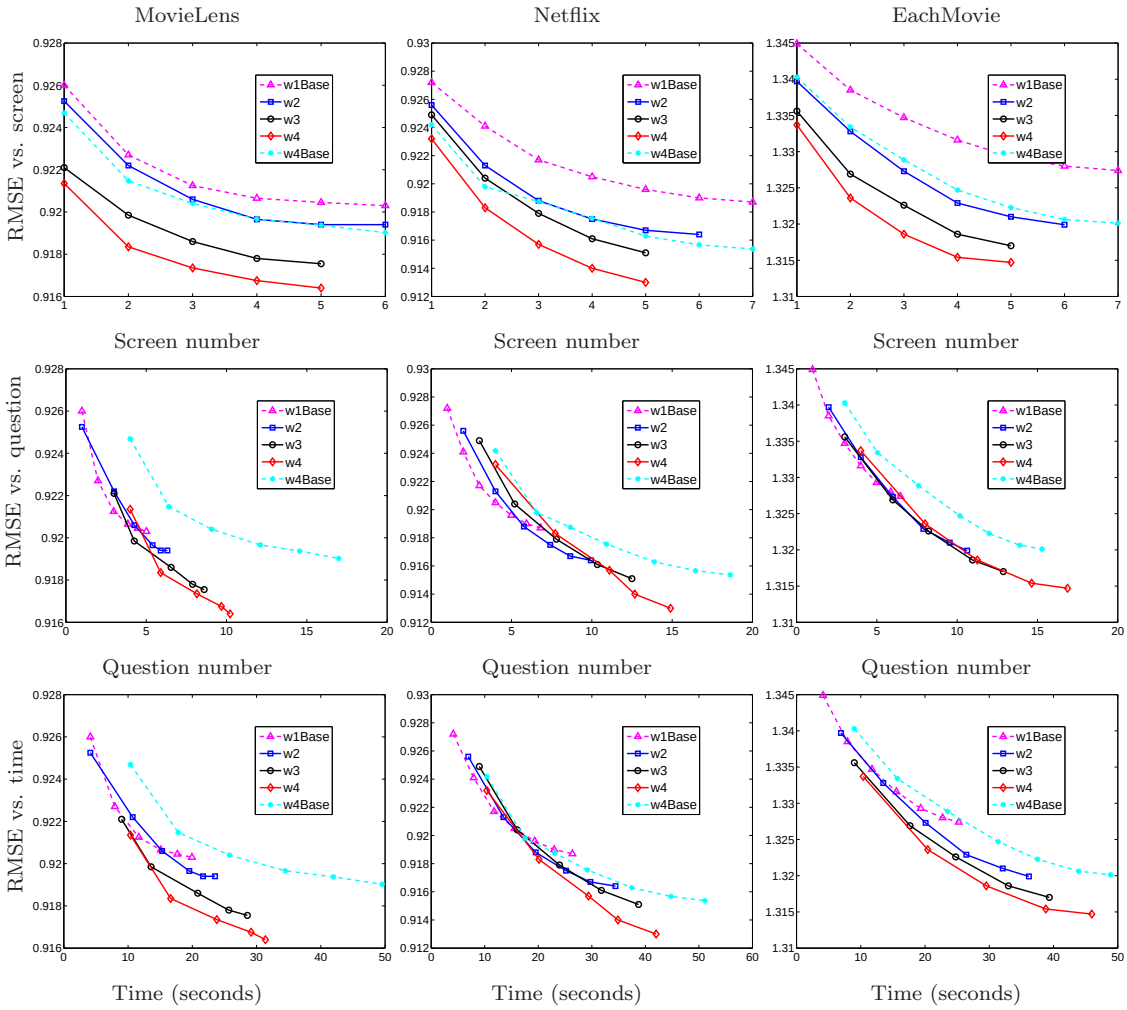


Figure 24: The prediction RMSE with respect to screen number, question number and user time on three data sets. We compare our methods asking 2, 3 and 4 questions with baseline $w1_{Base}$ asking single question and baseline $w4_{Base}$ asking 4 questions. For all methods, the prediction error measured in RMSE decreases as the screen number, question number increases. The first and second row shows that methods asking $2, 3, 4$ questions on each screen performs better than asking one question $w1_{Base}$. The time in the third row is the expected time users spend in the interview process measured from our user study.

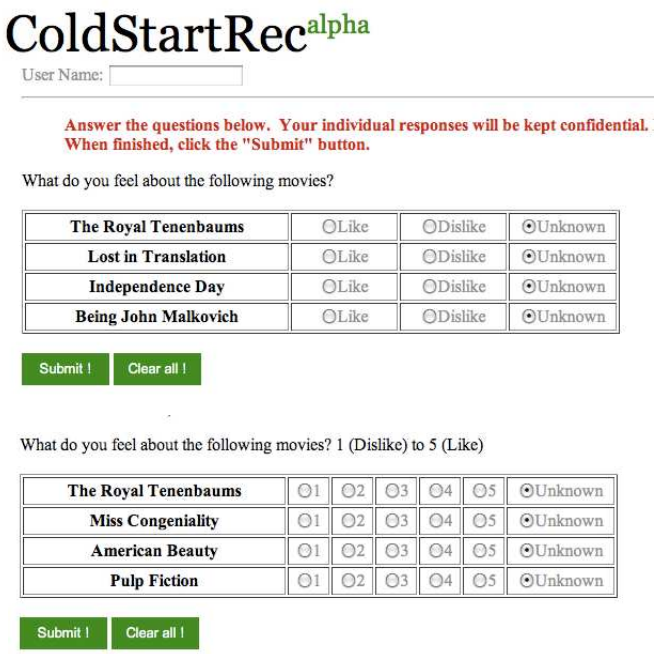


Figure 25: Interview web interface with 2 types of user input. Upper: binray. Bottom: rating scale.

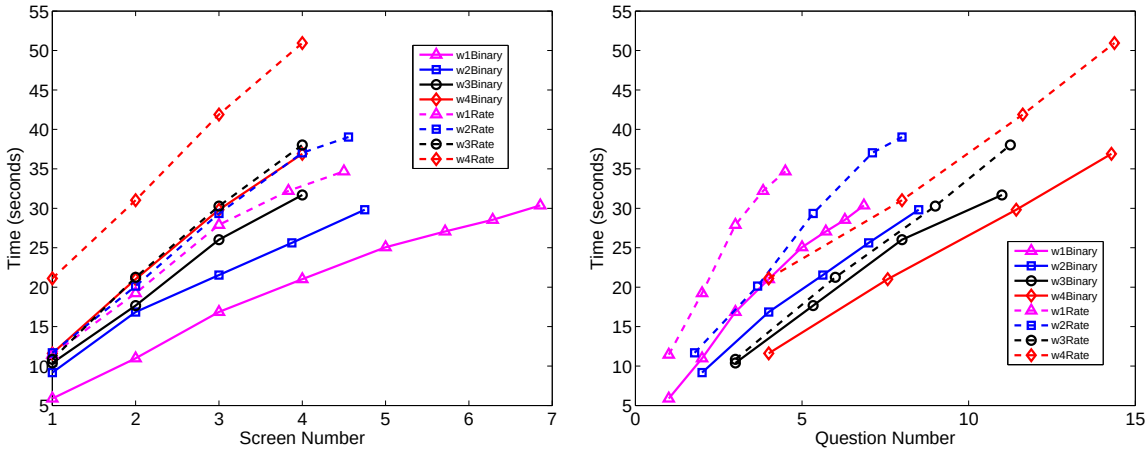


Figure 26: Time of the interview process with different question number per screen. The horizontal axis of left figure is the number of screen and the horizontal axis of right figure is the averaged number of accumulated questions.

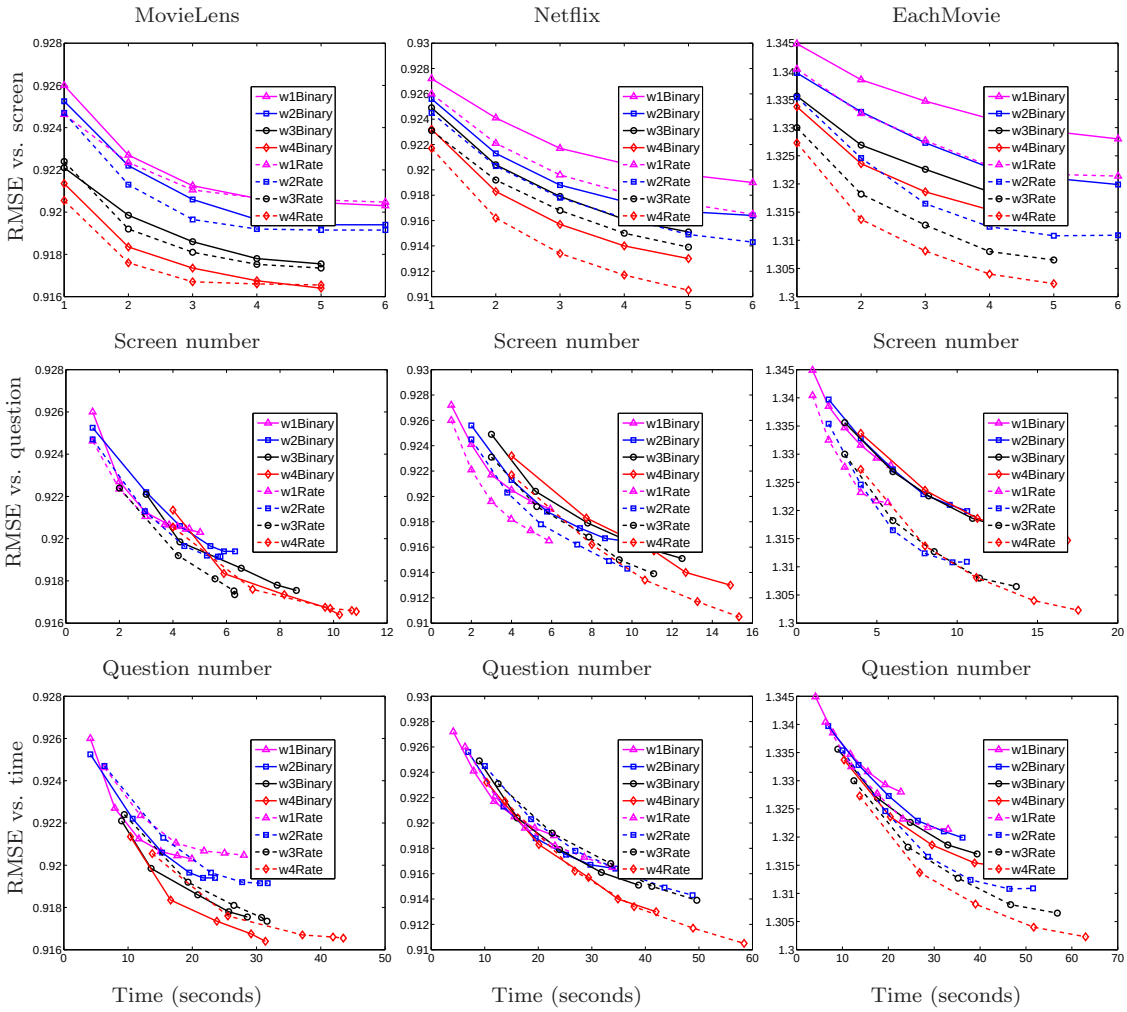


Figure 27: The prediction RMSE with respect to screen number, question number and user time on three data sets given different types of user answers. The *Binary* type expects users to answer *like* and *dislike*. $w2_{Binary}$, $w3_{Binary}$ and $w4_{Binary}$ are the tree models with 2, 3 and 4 questions on each screen respectively. In the *Rate* type, user chooses from a 5-star rating scale. $w2_{Rate}$, $w3_{Rate}$ and $w4_{Rate}$ are the tree models with 2, 3 and 4 questions on each screen respectively. Generally, rating scale provides richer information than binary answers, which leads to better prediction accuracy. On the other hand, rating scale may take more user time than binary answers.

CHAPTER VI

CONCLUSION

In this thesis, we focus on addressing problems of scalability for visualizing and modeling partial incomplete rankings. The main contribution is summarized as follows.

We presented a new distance measure for partial rankings with the following three properties: (1) metric, (2) emphasis on top ranks, and (3) computational efficiency. The distance in conjunction with multi-dimensional scaling is effective for visualizing partial rankings. We have explored the validity of the framework using a simulation study, which indicates that the weighted Hoeffding distance is more appropriate than Kendall's tau and more discriminative than the inverse measure. It is also more appropriate for MDS embedding than non-symmetric precision-recall measures such as NDCG. We also demonstrate the robustness of the proposed distance with respect to the choice of n and its efficient computation with complexity that is linear in the sizes of the ranked lists (assuming some quantities are precomputed offline). Experiments on search engine data have demonstrated that our framework can provide visual assistance in understanding the relationship between a search algorithm's ranked results.

We demonstrated a non-parametric estimator for partial incomplete rankings that is computationally tractable i.e., polynomial rather than exponential in n. The computation is made possible using generating function and combinatorial properties. We show experimentally that it performs similar to a state-of-the-art matrix factorization method and substantially outperforms other memory based methods (to which it is similar functionally). An advantage of our method is that its probabilistic nature makes it naturally suited for tasks that go beyond rank prediction such as finding

association rules, optimizing prediction for customized loss functions, and deriving confidence bounds.

We proposed a learning algorithm to construct a decision tree for preference elicitation from partial incomplete rankings. Specifically, we proposed a new algorithm to learn decision trees with multiple dimensions selected for each split, for applications in cold-start recommender systems. Based on L_1 regularization, a relaxation formulation is proposed to optimize for each split. The new tree has users with heterogeneous answers in the same node, therefore regressors are learned within each node based on previously given answers. Experiments show that the algorithm outperforms state-of-the-art approaches in terms of both the prediction accuracy and user cognitive efforts.

In our current density estimation framework, incomplete tied preference data are interpreted as randomly censored permutation data. From a practical perspective, robustness to departures from the assumptions must be considered, specifically random censoring and consistency. Previous work has demonstrated that the random censoring assumption may be violated as people tend to rate item that they feel strongly about more frequently than those for which they do not have strong feelings [67]. Such a tendency should not have a substantial negative impact on recommendations as attitudes toward polarizing items will be captured and the use of the notion of compatible sets encourages average ratings for infrequently rated movies. Secondly, while consistency may not hold in every case, scenarios can be devised which make it very likely, e.g. restricting attention to situations with very large n and only estimating probabilities over a very small number of items. Additionally kernel density estimators tend to flatten peaks and lift valleys, but the relative values of the probabilities will retain the same ordering likely mitigating the impact on the recommendations. In practice, the empirical performance observed over several data sets and in several

experiments indicates that any adverse effect based on departures from these assumptions produces an impact no larger than that experienced with other state-of-the-art approaches. To summarize, although these assumptions may not always hold, the practical impact is likely to be negligible.

For future work, an interesting extension is using the weighted hoeffding distance as an objective in learning to rank. In particular, the distance can be used to compare a ranked list to a ground truth of relevant and not-relevant documents or websites. Minimizing the distance is equivalent to optimizing top-k precision rankings. More general, we would like to explore research related to optimizing a function over permutation space. It would be useful to have efficient optimization routines in many situations such as finding a mean ranking in a population. Optimization in ranking space may be also utilized with decision tree frameworks for preference elicitation.

REFERENCES

[1] Agarwal, D. and Chen, B., "Regression-based latent factor models," in *Proc. of the ACM SIGKDD*, pp. 19–28, ACM, 2009.

[2] Alvo, M. and Cabilio, P., "On the balanced incomplete block design for rankings," *The Annals of Statistics*, vol. 19, no. 3, pp. 1597–1613, 1991.

[3] Alvo, M. and Cabilio, P., "Rank correlation methods for missing data," *The Canadian Journal of Statistics*, vol. 23, no. 4, pp. 345–358, 1995.

[4] Aslam, J. and Montague, M., "Models for metasearch," in *Proc. of ACM SIGIR Conference*, pp. 276–284, ACM, 2001.

[5] Baggerly, K. A., *Visual Estimation of Structure in Ranked Data*. PhD thesis, Rice University, 1995.

[6] Bar-Ilan, J., Keenoy, K., Yaari, E., and Levene, M., "User rankings of search engine results," *Journal of American Society for Information Science and Technology*, vol. 58, no. 9, pp. 1254–1266, 2007.

[7] Basu, C., Hirsh, H., and Cohen, W., "Recommendation as classification: Using social and content-based information in recommendation," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 714–720, 1998.

[8] Borg, I. and Groenen, P. J. F., *Modern Multidimensional Scaling*. Springer, 2nd ed., 2005.

[9] Boutilier, C., Zemel, R., and Marlin, B., "Active collaborative filtering," in *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pp. 98–106, 2003.

[10] Breese, J., Heckerman, D., and Kadie, C., "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. of Uncertainty in Artificial Intelligence*, 1998.

[11] Breiman, L., Friedman, J., Olshen, R., and Stone, C., *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[12] Busse, L., Orbanz, P., and Buhmann, J., "Cluster analysis of heterogeneous rank data," in *Proc. of the International Conference on Machine Learning*, pp. 113–120, ACM, 2007.

[13] Cayley, A., "A note on the theory of permutations," *Philosophical Magazine*, vol. 34, pp. 527–529, 1849.

[14] Chee, S. H. S., Han, J., and Wang, K., "Rectree: An efficient collaborative filtering method," in *Lecture Notes in Computer Science*, pp. 141–151, Springer Verlag, 2001.

[15] Cliff, N. and Keats, J., *Ordinal measurement in the behavioral sciences*. Lawrence Erlbaum Associates, 2003.

[16] Cohen, A. and Mallows, C., "Analysis of ranking data," tech. rep., Bell Laboratories, 1980.

[17] Cohen, W. C., Schapire, R. E., and Singer, Y., "Learning to order things," *Journal of Artificial Intelligence Research*, vol. 10, pp. 243–270, 1999.

[18] Connor, M. and Herlocker, J., "Clustering items for collaborative filtering," in *Proceedings of ACM SIGIR Workshop on Recommender Systems*, 2001.

[19] Cox, T. F. and Cox, M. A. A., *Multidimensional Scaling*. Chapman and Hall, 1994.

[20] Critchlow, D. E., *Metric Methods for Analyzing Partially Ranked Data*. Lecture Notes in Statistics, volume 34, Springer, 1985.

[21] Croft, B., Metzler, D., and Strohman, T., *Search Engines: Information Retrieval in Practice*. Addison Wesley, 2009.

[22] DeCoste, D., "Collaborative prediction using ensembles of maximum margin matrix factorizations," in *Proc. of the ICML*, 2006.

[23] Diaconis, P., *Group Representations in Probability and Statistics*. Institute of Mathematical Statistics, 1988.

[24] Diaconis, P., "A generalization of spectral analysis with application to ranked data," *Annals of Statistics*, vol. 17, pp. 949–979, Sept. 1989.

[25] Diaconis, P. and Graham, R. L., "Spearman's footrule as a measure of disarray," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 2, pp. 262–268, 1977.

[26] Fagin, R., Kumar, R., and Sivakumar, D., "Comparing top k lists," in *Proc. of ACM SODA*, 2003.

[27] Fligner, M. A. and Verducci, J. S., "Distance based ranking models," *Journal of the Royal Statistical Society B*, vol. 43, pp. 359–369, 1986.

[28] Fligner, M. A. and Verducci, J. S., "Multistage ranking models," *Journal of the American Statistical Association*, vol. 83, pp. 892–901, 1988.

[29] Fligner, M. A. and Verducci, J. S., "Posterior probabilities for a concensus ordering," *Psychometrika*, vol. 55, pp. 53–63, 1990.

[30] Freund, Y., Iyer, R., Schapire, R. E., and Singer, Y., "An efficient boosting algorithm for combining preferences," in *International Conference on Machine Learning*, 1998.

[31] Gabriel, K., "Biplot," *Encyclopedia of Statistical Science*, vol. 1, pp. 263–271, 1982.

[32] Golbandi, N., Koren, Y., and Lempel, R., "On bootstrapping recommender systems," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, pp. 1805–1808, ACM, 2010.

[33] Golbandi, N., Koren, Y., and Lempel, R., "Adaptive bootstrapping of recommender systems using decision trees," in *Proceedings of the fourth ACM international conference on Web search and data mining*, pp. 595–604, ACM, 2011.

[34] Goldberg, K., Roeder, T., Gupta, D., and Perkins, C., "Eigentaste: A constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, no. 2, pp. 133–151, 2001.

[35] Granka, L., Joachims, T., and Gay, G., "Eye-tracking analysis of user behavior in www search," in *Proc. of the ACM-SIGIR conference*, pp. 478–479, 2004.

[36] Gunawardana, A. and Meek, C., "Tied boltzmann machines for cold start recommendations," in *Proceedings of the 2008 ACM conference on Recommender systems*, pp. 19–26, ACM, 2008.

[37] Harpale, A. and Yang, Y., "Personalized active learning for collaborative filtering," in *Proc. of ACM SIGIR Conference*, pp. 91–98, ACM, 2008.

[38] He, L., Liu, N., and Yang, Q., "Active dual collaborative filtering with both item and attribute feedback," in *AAAI*, 2011.

[39] Heckerman, D., Chickering, D. M., Meek, C., Rounthwaite, R., and Kadie, C., "Dependency networks for inference, collaborative filtering, and data visualization," *Journal of Machine Learning Research*, vol. 1, 2000.

[40] Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J., "An algorithmic framework for performing collaborative filtering," in *Proc. of ACM SIGIR Conference*, 1999.

[41] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T., "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, vol. 22, pp. 5–53, January 2004.

[42] Hespanha, J., "Grpartition a matlab function for graph partitioning." http://www.ece.ucsb.edu/ hespanha.

[43] Hofmann, T., "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems*, vol. 22, no. 1, p. 115, 2004.

[44] Huang, J., Guestrin, C., and Guibas, L., "Fourier theoretic probabilistic inference over permutations," *Journal of Machine Learning Research*, vol. 10, no. May, pp. 997–1070, 2009.

[45] Huang, J., Kapoor, A., and Guestrin, C., "Efficient probabilistic inference with partial ranking queries," in *Proc. of Uncertainty in Artificial Intelligence*, 2011.

[46] Järvelin, K. and Kekäläinen, J., "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems*, vol. 20, no. 4, pp. 422–446, 2002.

[47] Jin, R. and Si, L., "A bayesian approach toward active learning for collaborative filtering," in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pp. 278–285, 2004.

[48] Joachims, T., Granka, L., Pan, B., Hembrooke, H., and Gay, G., "Accurately interpreting clickthrough data as implicit feedback," in *Proc. of the ACM-SIGIR conference*, pp. 154–161, 2005.

[49] Kendall, M. G., "A new measure of rank correlation," *Biometrika*, vol. 30, 1938.

[50] Kidwell, P. and Lebanon, G., "Triangular smoothing of partial and incomplete ranked data," in *Algebraic Methods in Probability and Statistics II* (Vienna, M. and Wynn, H., eds.), American Mathematical Society, 2010.

[51] Kidwell, P., Lebanon, G., and Cleveland, W. S., "Visualizing incomplete and partially ranked data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1356–1363, 2008.

[52] Kim, B., Lee, B., and Seo, J., "Visualizing set concordance with permutation matrices and fan diagrams," in *Interacting with Computers*, 2007.

[53] Kondor, R., Howard, A., and Jebara, T., "Multi-object tracking with representations of the symmetric group," in *Proc. of the International Conference on Artificial Intelligence and Statistics*, 2007.

[54] Koren, Y., "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Transactions on Knowledge Discovery from Data*, vol. 4, no. 1, pp. 1–24, 2010.

[55] Lakshminarayanan, B., Bouchard, G., and Archambeau, C., "Robust bayesian matrix factorisation," in *Proc. of the International Conference on Artificial Intelligence and Statistics*, 2011.

[56] Lawrence, N. D. and Urtasun, R., "Non-linear matrix factorization with gaussian processes," in *Proc. of the ICML*, 2009.

[57] Lebanon, G. and Lafferty, J., "Cranking: Combining rankings using conditional probability models on permutations," in *Proc. of the 19th International Conference on Machine Learning*, Morgan Kaufmann Press, 2002.

[58] Lebanon, G. and Lafferty, J., "Conditional models on the ranking poset," in *Advances in Neural Information Processing Systems 15*, pp. 431–438, The MIT Press, 2003.

[59] Lebanon, G. and Mao, Y., "Non-parametric modeling of partially ranked data," *Journal of Machine Learning Research*, vol. 9, pp. 2401–2429, 2008.

[60] Lee, D. and Seung, H., "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems*, 2001.

[61] Lemire, D. and Maclachlan, A., "Slope one predictors for online rating-based collaborative filtering," *Society for Industrial Mathematics*, vol. 5, pp. 471–480, 2005.

[62] Liggett, W. and Buckley, C., "Query expansion seen through return order of relevant documents," tech. rep., NIST, 2001.

[63] Mallows, C. L., "Non-null ranking models," *Biometrika*, vol. 44, pp. 114–130, 1957.

[64] Manmatha, R., Rath, T., and Feng, F., "Modeling score distributions for combining the outputs of search engines," in *Proc. of the 24th ACM SIGIR conference*, 2001.

[65] Marden, J. I., *Analyzing and modeling rank data*. CRC Press, 1996.

[66] Marlin, B., "Modeling user rating profiles for collaborative filtering," in *Advances in Neural Information Processing Systems*, 2004.

[67] Marlin, B. M. and Zemel, R. S., "Collaborative filtering and the missing at random assumption," in *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, 2007.

[68] Miyahara, K. and Pazzani, M. J., "Collaborative filtering with the simple bayesian classifier," in *Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence*, pp. 679–689, 2000.

[69] Miyahara, K. and Pazzani, M. J., "Improvement of collaborative filtering with the simple bayesian classifier 1," no. 11, 2002.

[70] Moffat, A. and Zobel, J., "Rank-biased precision for measurement of retrieval effectiveness," *ACM Transactions on Information Systems*, vol. 27, 2008.

[71] Paterek, A., "Improving regularized singular value decomposition for collaborative filtering," *Statistics*, vol. 2007, pp. 2–5, 2007.

[72] Pennock, D. M., Horvitz, E., Lawrence, S., and Giles, C. L., "Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach," in *Proc. of the Conference on UAI*, 2000.

[73] Popescul, A., Ungar, L. H., Pennock, D. M., and Lawrence, S., "Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments," in *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, 2001.

[74] Pu, P. and Chen, L., "User-involved preference elicitation for product search and recommender systems," *AI Magazine*, vol. 29, no. 4, p. 93, 2009.

[75] Radlinski, F., Szummer, M., and Craswell, N., "Inferring query intent from reformulations and clicks," in *Proceedings of the 19th international conference on World wide web*, pp. 1171–1172, ACM, 2010.

[76] Rashid, A., Albert, I., Cosley, D., Lam, S., McNee, S., Konstan, J., and Riedl, J., "Getting to know you: learning new user preferences in recommender systems," in *Proceedings of the 7th international conference on Intelligent user interfaces*, pp. 127–134, ACM, 2002.

[77] Rashid, A., Karypis, G., and Riedl, J., "Learning preferences of new users in recommender systems: an information theoretic approach," ACM, 2008.

[78] Rennie, J. and Srebro, N., "Fast maximum margin matrix factorization for collaborative prediction," in *Proc. of the International Conference on Machine Learning*, 2005.

[79] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J., "Grouplens: an open architecture for collaborative filtering of netnews," in *Proc. of the Conference on CSCW*, 1994.

[80] Rorvig, M. E., "Images of similarity: A visual exploration of optimal similarity metrics and scaling properties of TREC topic-document sets," *JASIS*, vol. 50, no. 8, pp. 639–651, 1999.

[81] Rubner, Y., Tomasi, C., and Guibas, L. J., "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, 2000.

[82] Salakhutdinov, R. and Mnih, A., "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *Proc. of the International Conference on Machine Learning*, 2008.

[83] Salakhutdinov, R. and Mnih, A., "Probabilistic matrix factorization," in *Advances in Neural Information Processing Systems*, 2008.

[84] Sarwar, B., Karypis, G., Konstan, J., and Reidl, J., "Item-based collaborative filtering recommendation algorithms," in *Proc. of the International Conference on World Wide Web*, 2001.

[85] Schein, A. I., Popescul, A., Ungar, L. H., and Pennock, D. M., "Methods and metrics for cold-start recommendations," in *Proc. of ACM SIGIR Conference*, pp. 253–260, ACM, 2002.

[86] Schmidt, M., *Graphical Model Structure Learning with L1-Regularization*. PhD thesis, The University of British Columbia, 2010.

[87] Smith, B. B., "Discussion of professor Ross's paper," *Journal of the Royal Statistical Society B*, vol. 12, no. 1, pp. 41–59, 1950.

[88] Spoerri, A., "Metacrystal: A visual interface for meta searching," in *Proceedings of ACM CHI*, 2004.

[89] Srebro, N., Rennie, J. D. M., and Jaakola, T. S., "Maximum-margin matrix factorization," in *Advances in Neural Information Processing Systems 17*, pp. 1329–1336, MIT Press, 2005.

[90] Stanley, R. P., *Enumerative Combinatorics*, vol. 1. Cambridge University Press, 2000.

[91] Sun, M., Lebanon, G., and Collins-Thompson, K., "Visualizing differences in web search algorithms using the expected weighted hoeffding distance," in *Proc. of the International World Wide Web Conference (WWW)*, 2010.

[92] Sun, M., Lebanon, G., and Kidwell, P., "Estimating probabilities in recommendation systems," in *Proc. of the International Conference on Artificial Intelligence and Statistics*, 2011.

[93] Thompson, G. L., "Generalized permutation polytopes and exploratory graphical methods for ranked data," *The Annals of Statistics*, vol. 21, no. 3, pp. 1401–1430, 1993.

[94] Thurstone, L. L., "A law of comparative judgement," *Psychological Review*, vol. 34, pp. 273–286, 1927.

[95] Ukkonen, A., "Visualizing sets of partial rankings," *Advances in Intelligent Data Analysis VII*, pp. 240–251, 2007.

[96] Ungar, L. H. and Foster, D. P., "Clustering methods for collaborative filtering," in *AAAI Workshop on Recommendation Systems*, 1998.

[97] Vaughan, L., "New measurements for search engine evaluation proposed and tested," *Information processing and management*, vol. 40, no. 4, pp. 677–691, 2004.

[98] Vogt, C. C. and Cottrell, G. W., "Fusion via a linear combination of scores," *Information Retrieval*, vol. 1, no. 3, pp. 151–173, 1999.

[99] Vucetic, S. and Obradovic, Z., "Collaborative filtering using a regression-based approach," *Knowledge and Information Systems*, vol. 7, pp. 1–22, 2005.

[100] Wand, M. P. and Jones, M. C., *Kernel Smoothing*. Chapman and Hall/CRC, 1995.

[101] Xue, G. R., Lin, C., Yang, Q., Xi, W. S., Zeng, H. J., Yu, Y., and Chen, Z., "Scalable collaborative filtering using cluster-based smoothing," in *Proc. of ACM SIGIR Conference*, 2005.

[102] Yemelichev, V. A., Kovalev, M. M., and Kravtsov, M. K., *Polytopes, Graphs, and Optimisation*. Cambridge University Press, 1984.

[103] Yu, K., Zhu, S., Lafferty, J., and Gong, Y., "Fast nonparametric matrix factorization for large-scale collaborative filtering," in *Proc. of ACM SIGIR Conference*, 2009.

[104] Zhou, K., Yang, S., and Zha, H., "Functional matrix factorizations for cold-start recommendation," in *Proc. of the ACM-SIGIR conference*, pp. 315–324, 2011.